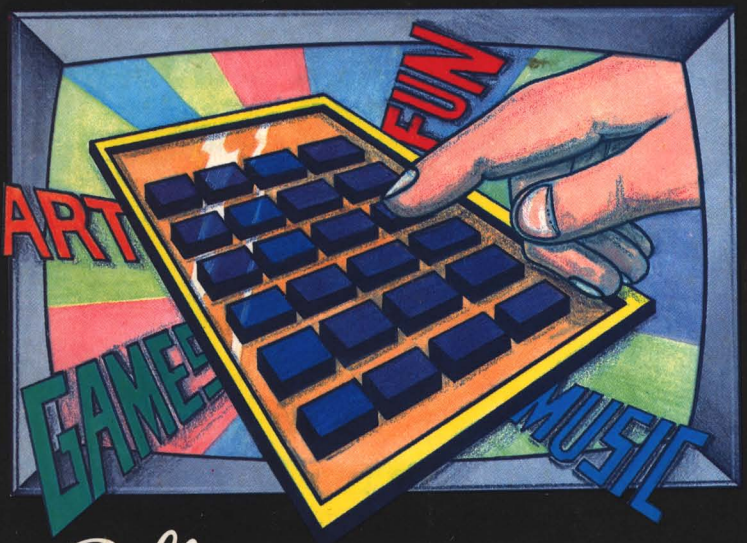
Bally®

PROFESSIONAL

Videocade™

CARTRIDGE



Bally® BASIC 6004

NOW WITH BUILT-IN AUDIO INTERFACE

AND ILLUSTRATED LEARNING COURSE

There's no easier way to learn about computers than with the new Bally BASIC system. This plug-in cartridge with built-in audio tape interface converts your ARCADE into a personal computer you can program yourself. The complete self-teaching instruction book will help you learn programming while you create computer games, electronic music, and video art.

Bally® BASIC

Bally® BASIC

By Dick Ainsworth with George Moses,
Jay Fenton and Brett Bilbrey

Bally® BASIC

Welcome to the world of computers. There are many versions of BASIC as well as several other computer languages. The term, BASIC, is an acronym for: Beginners' All-purpose Symbolic Instruction Code. Bally BASIC is a language designed to make computers and programming easier to understand. It is an expanded version of Li-Chen Wang's Palo Alto Tiny BASIC. Written by Jay Fenton, Bally BASIC allows you to program or create pictures and sounds accompanied by a full range of 256 color choices. Bally BASIC expands your computer by letting you program your own computer games, electronic music and video art.

These programming lessons, written by Dick Ainsworth, with George Moses, Jay Fenton and Brett Bilbrey, are an introduction to understanding and using your computer. You will learn how to enter programs so that you can see many of the things your computer can do. By experiencing how BASIC works, you will soon learn the special BASIC command words such as RUN and PRINT. Follow the lessons that interest you most. Then try using the computer to express your own ideas.

The special section on computer music, written by George Moses, shows how you can use your computer as a musical instrument. With these more advanced techniques you can use the computer's built-in sound synthesizer to play music in three-part harmony using your favorite sheet music, or you can program your own original compositions.

Bally® BASIC

CONTENTS

Operating Instructions

If you are using your Bally Arcade for the first time, please follow the directions in the OWNER'S MANUAL packaged with your unit. Connect your Bally Arcade to a black and white or color TV and play several of the games.

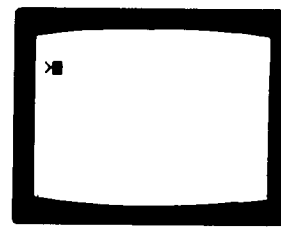
After you are familiar with your Arcade and know how it operates, use Bally BASIC and discover the enjoyment of having your own personal computer.

***REMOVE** the keypad overlay from its envelope in the front of this manual. (This envelope is a good place to store your overlay when you're not using it.)*

***INSERT** the Bally BASIC cassette in the cassette slot and press down firmly.*

***PLACE** the keypad overlay on the keypad.*

***RESET** your computer by pressing the RESET button next to the cassette. Your TV screen should look like this picture.*



CAUTION:

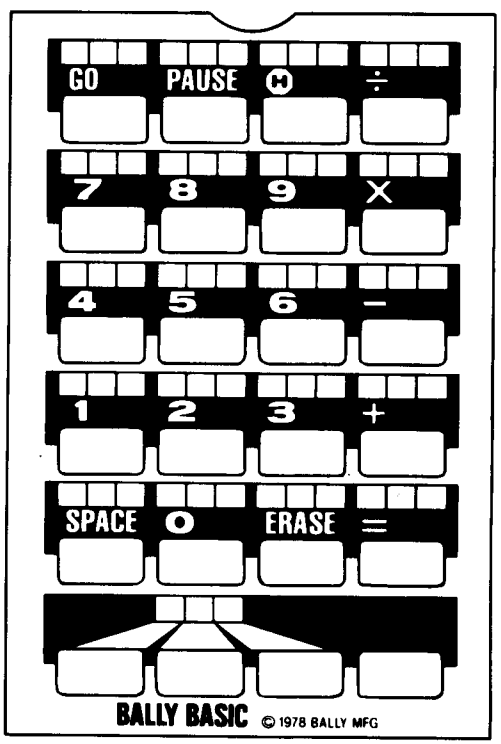
RESET erases your program. If you press this button by accident, you must enter your program again from the beginning.

EJECT causes your programming cassette to pop up, so you can remove it. Pressing the EJECT button accidentally will cause your program to stop. If this happens, push the cassette back into place, press RESET, and enter your program again.

STATIC that causes dots or "snow" on your TV screen or noise in the speaker can affect your computer. If static interrupts your program and causes it to stop, press RESET and enter the program again. If your body has a static-electricity build-up from walking across a carpet in a dry environment, discharge the spark before touching your computer. This is important. One high voltage spark such as this can seriously damage sensitive integrated circuits in your computer and should be carefully guarded against.

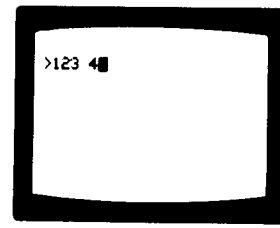
Using the Keypad

Numbers



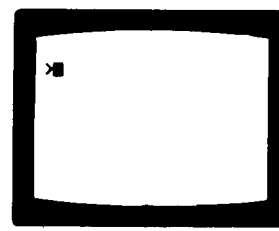
Your Bally BASIC keypad is used for three separate kinds of information: NUMBERS, LETTERS and WORDS. The WHITE numbers and symbols on your keypad are printed on your TV screen when you press those keys.

1
2
3
SPACE
4

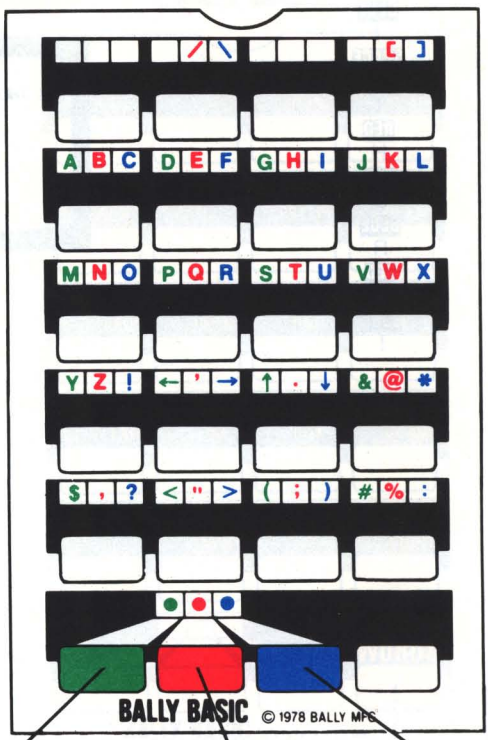


Now use the ERASE key to remove the numbers from the screen.

ERASE
ERASE
ERASE
ERASE
ERASE



Letters



The GREEN shift key selects characters on the left.

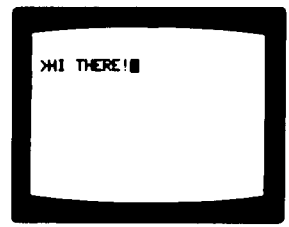
The RED shift key selects characters in the center.

The BLUE shift key selects characters on the right.

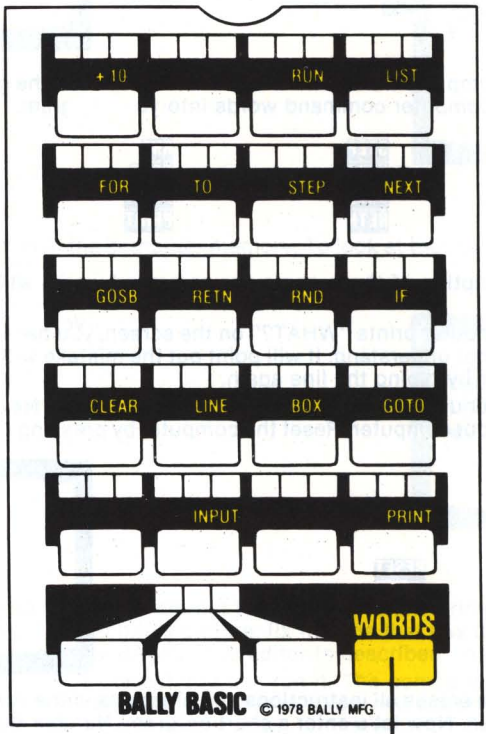
To print a letter or character on your TV screen, use the shift key in the same color. First press either the GREEN, RED or BLUE shift key to select a letter of the corresponding color. Then press the key that is under the letter you want to print.

Don't hold down the shift key when you press a letter or character key. Use one discrete keystroke on each key.

RED
H
BLUE
I
SPACE
RED
T
RED
H
RED
E
BLUE
R
RED
E
BLUE
!



Words



The GOLD shift key selects the WORDS printed in GOLD.

Words that the computer understands are printed on the keypad in GOLD. Press the WORDS key, and then press the key under the word you want to print on your TV.

WORDS
BOX
WORDS
STEP
WORDS
FOR

Never spell out computer words like B-O-X or S-T-E-P — use the gold WORDS shift key to enter these computer command words into your program.

RUN
LIST
FOR
TO
+ 10

STEP
NEXT
GOSB
RETN

RND
GOTO
INPUT
PRINT

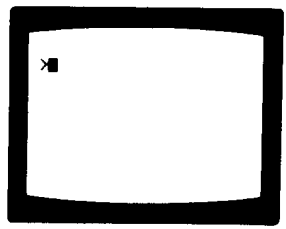
IF
CLEAR
LINE
BOX

A complete description of these computer command words will be discussed in this book.

NOTE: If your computer prints "WHAT?" on the screen, you have typed something the computer does not understand. It will point out the mistake with a question mark (?). Correct the error by typing the line again.

You can now print numbers, letters and words on the screen. Next you will learn to put programs into your computer. Reset the computer by pressing the RESET button.

RESET



The RESET button erases all instructions and programs in the computer's memory and clears the screen. Now let's enter a short program. Number the first instruction 10. Use the WORDS key to enter PRINT, and then spell out "HELLO!" The quotation marks are important because only the letters you put between them will be printed.

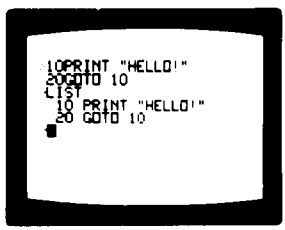
10PRINT "HELLO!"
GO

The GO key acts like a carriage return on a typewriter and moves you to the next line while entering the instruction you have just typed into the memory. Line numbers are used to tell the computer what to do next. NOTE: If you fill up a line with 26 characters and spaces the computer will automatically shift and start printing on the line below it. Now add the second instruction to your program and number it 20. Notice that GOTO is one word. Press the GO key to end the line.

20GOTO 10
GO

Now the program is in the computer memory. To look at the complete program, ask the computer to LIST it.

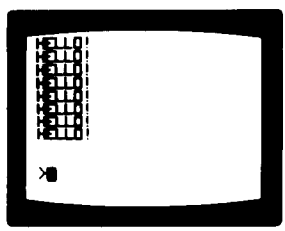
LIST
GO



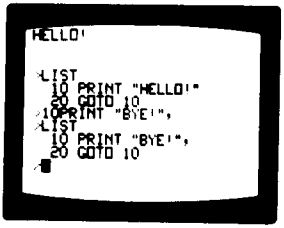
Notice how line 20 tells the computer to go to line 10 and print "HELLO!" again. If your TV screen doesn't match this picture, fix the line that has the mistake by using the EDITOR feature in Bally BASIC. (The editor is described on page 16.)

Now, if your program is correct you can run it. The computer will print the word "HELLO!," go back to the beginning of your program and start over. To stop the program, press the halt key, H, and hold it down until the computer halts.

RUN
GO
H

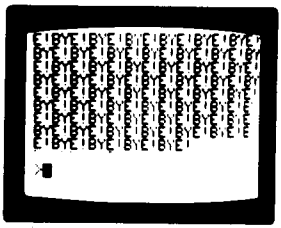


Now LIST your program again. You need to know how to change a program so that you can fix a mistake or make your program do something different. To change an instruction enter the number of the line you want to change and then enter the new instruction. The computer will replace the old line of the same number with the new line. Now enter this new instruction to replace line 10 in your program. Don't forget the comma at the end!



10PRINT "BYE!",
GO
LIST
GO

If your program matches the example, RUN it and see what it does. Use the HALT key **H** to stop.

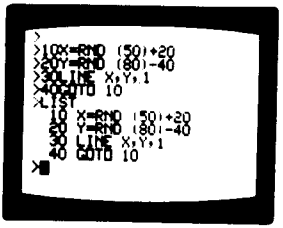


RUN
GO
H

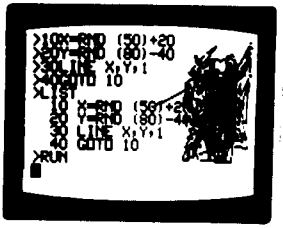
Now enter and run this longer program. After RESET and at the end of each line, press the WORDS key before you press the GO + 10 key. Your computer will number the lines 10, 20, 30 and 40 automatically. Be careful to use the letter x, not the multiplication sign (×).

RESET
10X = RND(50) + 20
20Y = RND(80) – 40
30LINE X,Y,1
40GOTO 10

GO
LIST
GO



Check your program to make sure it matches the example, then run it.



RUN
GO

This program will draw a random line that fills the right side of your screen. You can use the PAUSE key to stop the program; press any key to start it again. You will see how to program random events and create pictures in the programming course that begins on the following pages.

In the PROGRAMS section, pages 71 - 97, are many programs you can select, enter into your computer, and run. Remember to press RESET before each program and to press GO after each line. If you want to number lines automatically, press WORDS and GO.

THE AUDIO CASSETTE INTERFACE

The audio socket on the lower-right corner of your Bally BASIC cartridge is your connection port to the interface between your computer and any cassette tape recorder. This will allow you to save any program in the computer's memory on cassette tape and to input the program from tape to memory in less than 20 seconds. The following commands relate to tape storage and retrieval of Bally BASIC programs.

:PRINT

The :PRINT command causes the stored program, the screen image, the values stored in the @( ) and *( ) arrays, and the values of all variables to be output to tape. This process will take between 10 and 20 seconds. As only one jack is provided on the BASIC cartridge, it is necessary for the user to manually connect the audio cable to the MIC jack of the cassette recorder. Have your recorder running in the RECORD mode, type in :PRINT and press GO. When the cursor reappears, the computer is done writing your program to tape.

The recording will consist of a 3 second leader tone, then the data block, followed by a ½ second trailer tone.

:INPUT

To load programs, use :INPUT. This will retrieve the program, the screen image, the arrays and variables from audio tape. It is necessary to "cue up" the audio tape on the three second leader tone and switch the cable over to the EARPHONE jack on your tape recorder before loading. A light emitting diode (LED) is provided on the lower left corner of your Bally BASIC cartridge to aid in checking the playback level of your cassette recorder. The volume should be set above the level that causes the LED to glow steadily. When the tape is cued, type :INPUT GO and press PLAY on your tape recorder. When the cursor reappears your program is loaded.

:LIST

The :LIST command has been designed to perform a verify function. It scans a digital recording and checks the sum of the bits in the recorded program against the sum of the bits in the program currently in memory. The :LIST function is to be performed just after writing your program to tape with the :PRINT command, while your program is still in the computer. This allows you to check the integrity of your recording without damaging the program. If :LIST finds an error, a question mark is printed just before the cursor when it returns. If problems arise, check the playback level of your recorder, or rewrite the program to tape if necessary.

EDITOR

You can use the Bally BASIC editor to change a line in your program.

1. Type the line number of the line you wish to correct.
2. Press the PAUSE key once to recall the next character from the line in storage. Repeated pressing of PAUSE will scan across the stored line.
3. A character drawn by PAUSE looks to the computer just like a character entered directly from the keypad. This means ERASE can be used to discard unwanted characters from the stored line. Additions can also be added from the keypad, intermixed as desired.
4. If PAUSE runs off the end of a stored line, it will have the same effect as pressing the GO key. The key sequence WORDS—SPACE will also activate this feature.

THE TRACE FEATURE

Now that you have a program in the memory of your Arcade,let's try out the TRACE feature. TRACE is a self-teaching function built into Bally BASIC to let you see which statements in your program are operating while you watch the program run. A statement by statement TRACE listing may be obtained while the program is executing by holding down the WORDS key and the BLUE key at the same time. Each statement is listed completely before it is executed.

REVIEW

Check these steps to make sure you understand how to operate your computer and enter programs.

1. Insert your Bally BASIC Programming Cassette and put the keypad overlay in place.

2. Press RESET (next to cassette). This erases any old programs.

3. Enter each instruction and press GO. (If you press WORDS and GO + 10 after each line, the computer will automatically add 10 and print a new line number for you.)

4. LIST the program and check each instruction carefully. The PAUSE key lets you pause when listing long programs.

5. If there are any mistakes, correct the line using the EDITOR feature or, if you wish, enter the instruction again using the same line number. To remove an instruction completely, reenter its line number and press GO.

6. When your program matches the example, press RUN and GO.

Next Step...

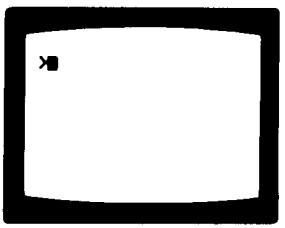
Now you can continue with Lesson One and learn how to write your own programs, or you can go to the PROGRAMS section of this manual and try any of the programs you like.

LESSON 1
PRINTING, COUNTING AND LOOPS

Before you begin these lessons, please read and understand the OPERATING INSTRUCTIONS. They begin on page 6 and show you how to enter, list and run programs on your computer.

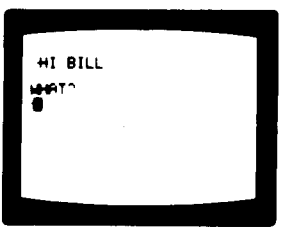
Learning how to write your own programs is not hard at all. Soon you will be able to have your computer play your own games, music and video art. Begin by writing a short program. First clear the computer's memory with the RESET button:

RESET



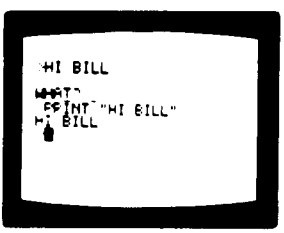
Now spell out HI and your name. Use the color shift keys as described on page 8 . To type the letter H, for example, press the red shift key and then the key under the red letter H. After you have finished typing, press GO.

HI BILL
GO



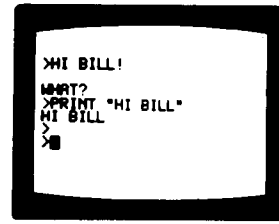
The computer prints "WHAT?" on the screen because it doesn't know what you said. The words "HI" and "BILL" are not words your computer understands. Instead, use the WORDS shift key to enter the word PRINT. Then spell out "HI BILL." Don't forget the quotation marks.

PRINT "HI BILL"
GO



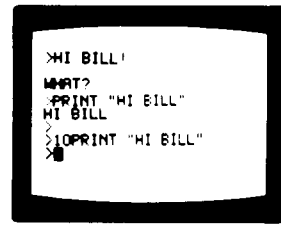
PRINT is one of the special words your computer understands. When you pressed GO, the computer followed your instruction and printed the words between the quotation marks. Now press GO again and see what happens:

GO



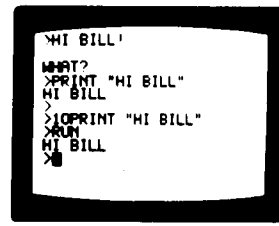
You can't print these words a second time because the computer doesn't remember what to do. To have your computer remember your instruction, just give it a line number. Number your instruction 10 and enter it again.

10PRINT "HI BILL"
GO



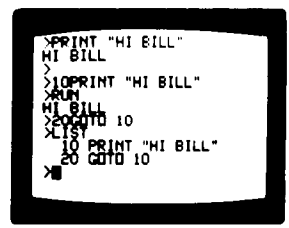
Now you have a one-line program in the computer memory. You can run this program as many times as you like. To run your program, use the WORD shift key and enter RUN.

RUN
GO



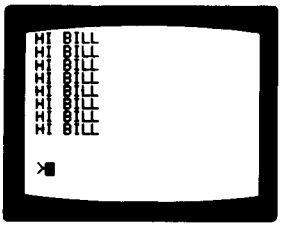
Add a second instruction to your program and number it 20. Then list your program.

20GOTO 10
GO
LIST
GO



Here's what your program will do. The computer will print HI BILL, go back to the beginning again, print HI BILL, and continue until you press the HALT key.

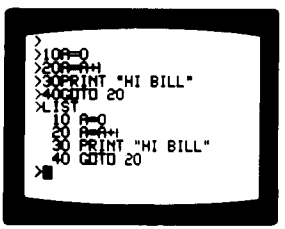
RUN
GO
H



Press and hold H until the computer stops.

How many times did you run your program? There is an easy way to find out. Make a counter to keep track of the number of times it ran. Reset your computer, then enter and list this new program.

RESET
10A = 0
20A = A + 1
30PRINT "HI BILL"
40GOTO 20
GO
LIST
GO



This program uses the letter A as a counting variable. Here's what happens when you run the program.

In line 10 the computer sets A equal to 0.

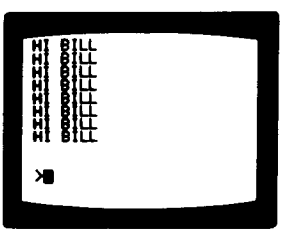
In line 20 the computer adds 1 to the number stored in A.

In line 30 the computer prints whatever is between the quotation marks.

In line 40 the computer goes back to line 20, adds one more to the number stored in A, and repeats.

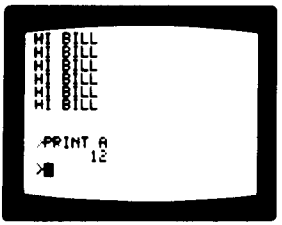
Run your program and print "HI BILL" about a dozen times. Then press and hold the halt (H) key.

RUN
GO
H



Each time the computer printed "HI BILL," it added 1 to the variable A. To find out how many times your program ran, print the value of the variable A.

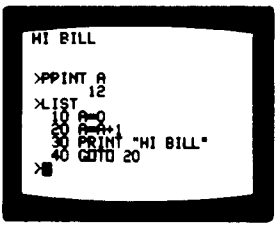
PRINT A
GO



In the example shown here the program ran 12 times and A = 12.

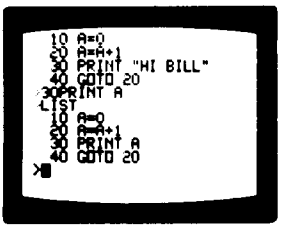
When you say print "A" the computer prints the letter A. When you say PRINT A the computer prints the value of A, or the number in the counting variable A. You can also use any other letter as a variable. Now list your program again.

LIST
GO



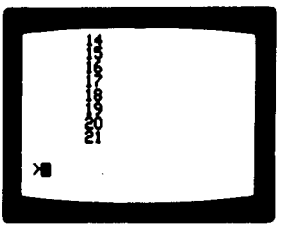
Instead of printing "HI BILL" over and over, you can print the number in the counting variable A. Change line 30 by entering the same line number and adding the new instruction, then list your program.

30PRINT A
GO
LIST
GO



The computer always lists the latest version of your program. This time, before you run your program, try to figure out what it will do. Then run it and see if you were right.

RUN
GO
H



The computer sets the variable A to zero in line 10. In line 20, 1 is added to A. Next, in line 30, the computer loops back to line 20 and repeats.

COUNTING LOOP

```
10A=0
20A=A+1
30PRINT A
40GOTO 20
```

WHAT THE COMPUTER DOES

```
10A=0
20A=1
30PRINT 1
40GOTO 20

20A=2
30PRINT 2
40GOTO 20

20A=3
30PRINT 3
40GOTO 20
```

Until you press halt H

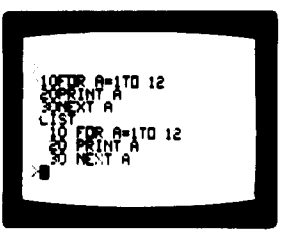
Programs that repeat are called loops. Another way to program a loop is with the words FOR and NEXT. FOR, TO and NEXT are computer words. Use the WORDS shift key to enter them just as you enter RUN, PRINT, LIST and all other Bally BASIC computer words.

RESET your computer to erase the counting loop and enter this program.

RESET

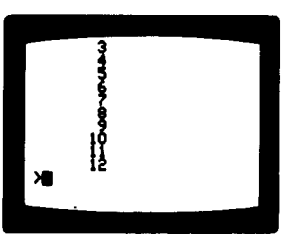
```
10FOR A=1TO 12
20PRINT A
30NEXT A
```

GO
LIST
GO



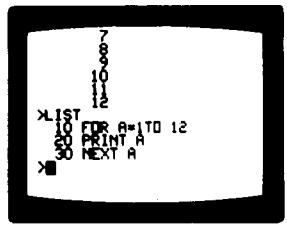
In line 10 the computer sets the variable A to 1. In line 20, A is printed. The word NEXT in line 30 means add 1 to A and loop back to the word FOR. NEXT A replaces A=A+1 and GOTO 20 which were used in the last program. Run your program and print the number in A as the A goes from 1 to 12.

RUN
GO



This time the program loop stopped automatically at 12. List your program again.

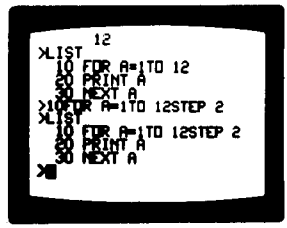
LIST
GO



FOR/NEXT LOOP	WHAT THE COMPUTER DOES
10FOR A = 1TO 12	10A = 1
20PRINT A	20PRINT 1
30NEXT A	30A = 2;GOTO 20
	20PRINT 2
	30A = 3;GOTO 20
	20PRINT 3
	30A = 4;GOTO 20
	Until A = 12

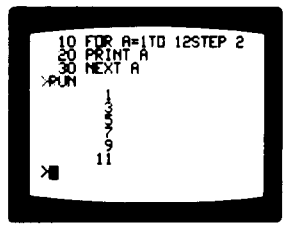
The FOR/NEXT loop adds 1 to the variable A. You can also add 2, 3, or any other number to a variable. Change line 10 to count by 2's.

10FOR A = 1TO 12STEP 2
GO
LIST
GO



Now run your program and see if it prints all the odd numbers between 1 and 12.

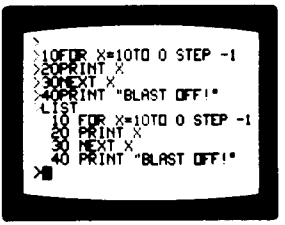
RUN
GO



You could also change line 10 and print all the tens from one to one hundred or all the leap years since your birthday. You can even step backwards by using negative numbers. Erase the program now in the computer with RESET, and enter this new program.

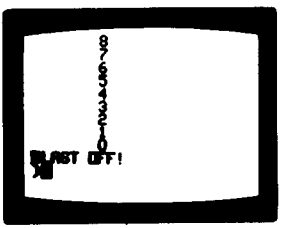
RESET

10FOR X = 10 TO 0STEP −1
20PRINT X
30NEXT X
40PRINT "BLAST OFF!"
GO
LIST
GO



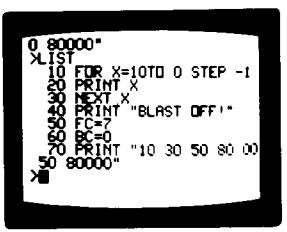
Now run your program. You're at 10 seconds and counting!

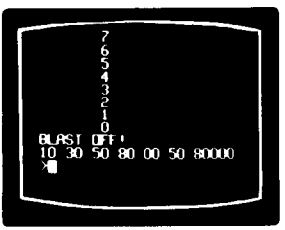
RUN
GO



Now for some fun to end your first lesson. Add the following three lines to your program. Lines 50 and 60 change the foreground color (controlled by the two-letter variable FC) and the background color (controlled by the two-letter variable BC) and line 70 plays a tune at the end. Notice that line 70 is now longer than the screen. Just continue entering numbers. The computer will scroll the entire screen up one line to make room for the new bottom line automatically whenever the maximum of 11 lines of printing is already on the screen. Enter a multiplication sign (on the key marked "NEXT") as shown below with (×).

50FC = 7
60BC = 0
70PRINT "100300500 × 100000
0500 × 100000000"





The improved program should play a tune at the end. Try it!

You will learn all about colors in Lesson 8, and music is explained in Lesson 5. The remaining lessons are no more difficult than the one you have just completed, so feel free to skip ahead to whatever subject is most interesting.

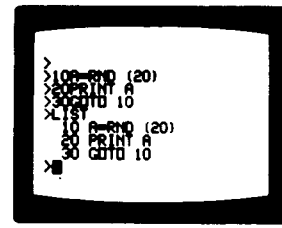
LESSON 2
RANDOM NUMBERS, INPUTS AND WHAT IF?

It's often handy to have your computer pick out numbers at random. Here's a program that selects random numbers between one and twenty.

RESET

```
10A = RND(20)
20PRINT A
30GOTO 10
```

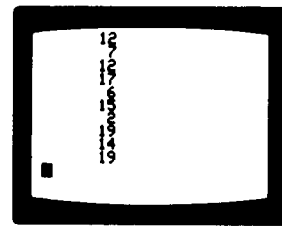
GO
LIST
GO



In line 10 the computer will set the variable A equal to a random number between one and twenty. In line 20 the computer prints the value of A. Line 30 sends the computer back to line 10. The computer continues picking a random number, printing it, and looping back to the beginning of the program.

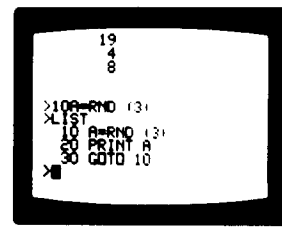
The numbers this program selects are different each time, so don't expect your numbers to match the example.

RUN
GO



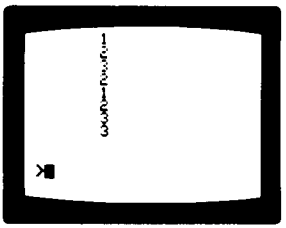
Now change line 10 to put random numbers from one to three into the variable A.

H
10A = RND (3)
GO
LIST
GO



Now run your program and let it list a few numbers.

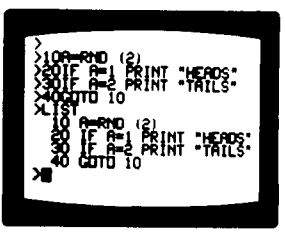
RUN
H



IF is a computer word that lets you check and see whether or not an expression is true. Enter this program:

RESET

10A = RND (2)
20IF A = 1PRINT "HEADS"
30IF A = 2PRINT "TAILS"
40GOTO 10
GO
LIST
GO



First, the computer picks a random number that is either 1 or 2 and sets the variable A equal to the number picked. If A = 1, the computer prints "HEADS," and if A = 2, the computer prints "TAILS." Then the computer goes back to line 10 again, and the loop continues.

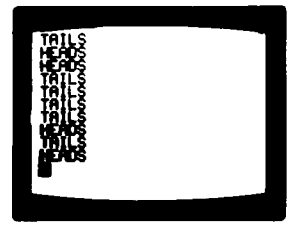
The IF command tests the value of the expressions IF A = 1 and IF A = 2. If either one of these expressions are true, the value of the expression is 1. If the expression is false, its value is 0. When an IF statement is true the rest of the statement will be executed. So IF A is equal to 1 in line 20, the computer will execute the command to print "HEADS." IF A is equal to any number except 1, the expression is false with a value of 0. The rest of the statement will be skipped and execution continues at the next statement. The value of any expression (1 or 0, true or false) can be tested with a simple immediate command:

PRINT A = 3
GO

If the statement is true, the computer will print "1." If the statement is false, the computer will print "0."

In the program you just typed into your computer you are using RND (2) to select a number. Depending on whether the number is 1 or 2, the computer prints either "HEADS" or "TAILS." Now run the program and see if heads or tails come up more often.

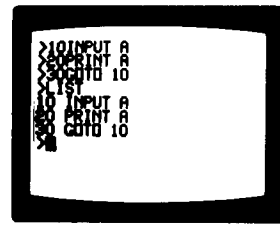
RUN
GO



Another way to change numbers in a program is to enter them yourself with INPUT. When the computer comes to an INPUT command in the program, it stops and waits for you to enter a number from the keypad, stores that number in a variable, and continues running the program. A letter variable, in this case, A, after the word INPUT tells the computer which variable to use.

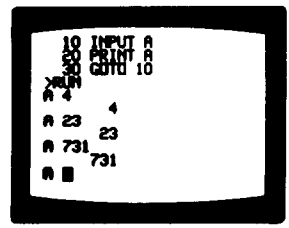
RESET

10INPUT A
20PRINT A
30GOTO 10
GO
LIST
GO



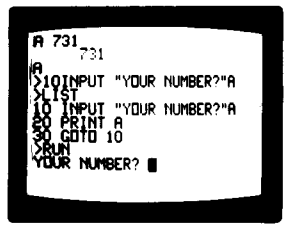
When you run the program, the computer will stop and wait for you to input a number. After you input a number, the computer prints the number you entered and asks for a second number. Follow the suggestions or try your own.

4
GO
23
GO
731
GO



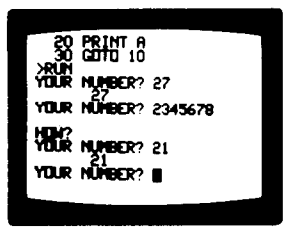
The computer prints "A" to remind you that your input will be stored in the variable A. You can have the computer remind you in other ways, too. Try this change in the program:

H
10INPUT "YOUR NUMBER?"A
GO
LIST
GO
RUN
GO



Now enter these numbers:

27
GO
2345678
GO
21
GO

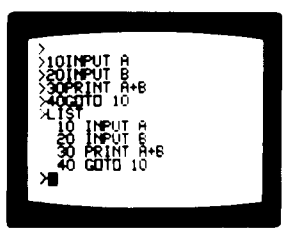


The largest number your computer's memory can hold is 32767. You just saw what happens when you input a number larger than that. The computer will ask WHAT? when it doesn't understand you. It will ask HOW? when it understands but can't do what you requested.

You have been using INPUT to set a variable equal to the number you type. This next program uses two variables and inputs numbers for A and B.

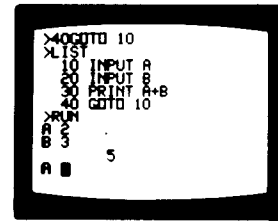
RESET

10INPUT A
20INPUT B
30PRINT A+B
40GOTO 10
GO
LIST
GO



The variable A is set equal to the first number you enter and the variable B is set equal to the second number. These two numbers are stored in the variables A and B. The computer prints their sum (A + B) and loops back to the beginning of your program. Try adding these numbers together then try some of your own.

RUN
GO
2
GO
3
GO

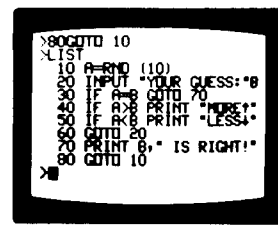


Input lets you put numbers into the computer and RND has the computer pick numbers at random. Now you can combine these and build a guessing game.

RESET

```
10A = RND(10)
20INPUT "YOUR GUESS:"B
30IF A = BGOTO 70
40IF A►BPRINT "MORE"
50IF A◄BPRINT "LESS"
60GOTO 20
70PRINT B,"IS RIGHT!"
80GOTO 10
```

GO
LIST
GO



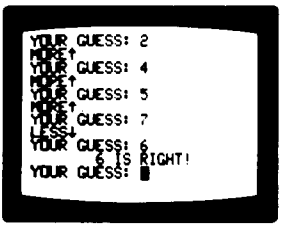
This program is longer than your others so we'll look at it step-by-step. First the computer picks a random number between one and ten and stores it in A. When you try to guess the number, your input is stored in B.

Now there are three things that can be true. If A = B then your guess is right. The computer goes to line 70 and prints your answer and the words "IS RIGHT!" If A is larger than B, A►B, then your guess is too small and the computer prints "MORE." If A is smaller than B, A◄B, then your guess is too big and the computer prints "LESS."

There are two loops in this program. If A = B the computer goes to line 70, prints the number you picked and the words "IS RIGHT!" and then loops back to the beginning to start a new game. If you didn't get the right answer the computer loops back to line 20 so you can try again.

HERE'S HOW A SAMPLE RUN MIGHT LOOK:

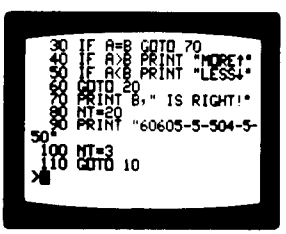
RUN
GO
2
GO
4
GO
5
GO
7
GO
6
GO



You can change line 10 to A = RND (100) and make the game harder, or add a counter to keep track of the number of guesses it took. Any of the words inside the quotation marks, like "MORE" can be changed to say whatever you want. Before you try your game on your friends, learn how to win every time. When the computer asks for your guess, just enter the letter A.

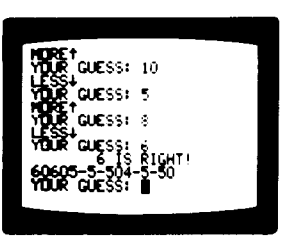
Here's a program add-on that you will like. Enter the line numbers as shown and the computer will add them to the program in the right order.

H
80NT = 20
90PRINT "60605 – 5 – 504 – 5 – 50"
100NT = 2
110GOTO 10
GO
LIST



Now try the guessing game again and be ready for a surprise when you get the answer right!

RUN
GO



LESSON 3
SUBROUTINES

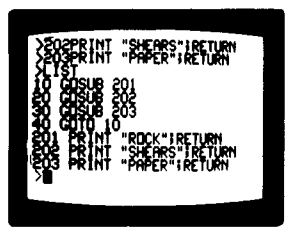
It can be very helpful to create a subroutine whenever you wish to repeat an action several times in a program. Any instruction or group of instructions can be used. These instructions are "called" in the program with the computer word GOSUB. When the computer reads the instruction GOSUB 1000, for example, it transfers to line 1000 and follows the instructions listed there. When the computer reads the word RETURN, it returns to the instruction following the GOSUB command.

This program prints the words ROCK, SHEARS and PAPER several times. To avoid having to write these same print instructions over and over, we will use GOSUB and RETURN to create a subroutine.

These lines form the first part of the program.

RESET

10GOSUB 201
20GOSUB 202
30GOSUB 203
40GOTO 10
201PRINT "ROCK";RETURN
202PRINT "SHEARS";RETURN
203PRINT "PAPER";RETURN

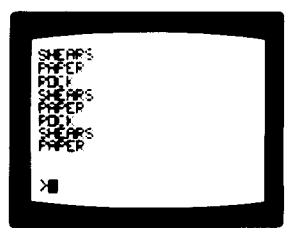


GO
LIST
GO

Here is what will happen. When the computer reads line 10, it will jump to the subroutine at line 201 and continue until it reaches the word RETURN. Then the computer will return to line 20 and continue. The computer will also go to subroutines as directed in lines 20 and 30, returning when it reads the word RETURN.

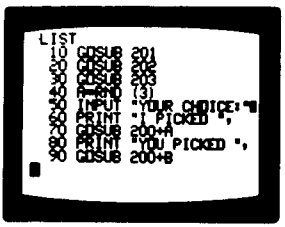
In line 40, the GOTO instruction tells the computer to go back to line 10 and start the program over again. Now RUN this part of your program.

RUN
GO
H

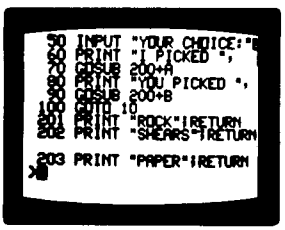


When you add the following lines to your program it will be too long to fit on the screen, but it will scroll up after the maximum of 11 lines of 26 characters occupy the screen, allowing you to keep typing on the bottom line.

```
40A = RND (30)
50INPUT "YOUR CHOICE:"B
60PRINT "I PICKED",
70GOSUB 200 + A
80PRINT "YOU PICKED",
90GOSUB 200 + B
100GOTO 10
GO
LIST
GO
```



After you have entered these additional instructions, list the program. While it is being listed on the screen, press PAUSE to stop the listing so you can read it. To continue the listing, press any key. A feature called SCROLL MODE CONTROL has been built into your Bally BASIC to allow more user control of the scrolling process which occurs when printing reaches the bottom of the screen.



SCROLL MODE CONTROL

The two-letter SCROLL MODE variable is SM, which can be set to any one of five values:

SM = 0 Scrolls conventionally.
SM = 1 Suppresses scrolling. Cursor stays at bottom.
SM = 2 Holds cursor at screen bottom, clears after each carriage return.
SM = 3 Clears the screen and resets cursor to screen top.
SM = 4 AUTO-PAUSE. Allows the listing to fill the screen to the bottom line, at which time the computer will go into the pause mode until you touch any key. Then the screen will clear and start printing from the top line down, automatically pausing again when the screen is full. This is the ideal mode for viewing your program listings "one page at a time."

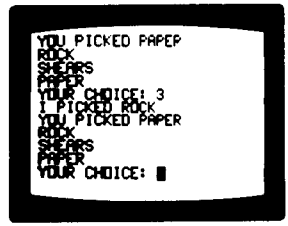
Going back to our program, here's what you've added. In line 40 the computer will select 1, 2, or 3 at random and set the variable A to this value. In line 50 the computer will ask for your choice (1, 2, or 3) and set B equal to the number you input.

Line 60 prints "I PICKED" and line 70 goes to a subroutine at line number 200 + A. If A = 1 the computer will GOSUB to line 201. If A = 2, it will GOSUB to line 202. And if A = 3, it will GOSUB to line 203. Depending on the value of A, "ROCK," "SHEARS," or "PAPER" will be printed after the words "I PICKED." And the comma ending lines 60 and 80 tells the computer to keep printing on the same line.

Lines 80 and 90 use the same GOSUB feature to print your selection. Line 100 loops the program back to the beginning.

RUN your program and INPUT 1, 2 or 3 to select ROCK, SHEARS, or PAPER.

```
RUN
GO
1
GO
2
GO
3
GO
```

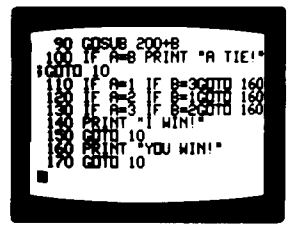


Now you can play ROCK, SHEARS, PAPER with your computer. The rules are:

ROCK breaks SHEARS
SHEARS cut PAPER
PAPER wraps ROCK

You can add more instructions to the program so that the computer will tell you who won. Halt the program and add that feature with these lines:

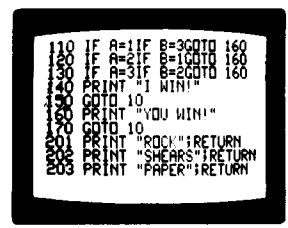
```
H
100IF A=BPRINT "A TIE!";GOTO 10
110IF A=1IF B= 3GOTO 160
120IF A=2IF B=1GOTO 160
130IF A=3IF B=2GOTO 160
140PRINT "I WIN!"
150GOTO 10
160PRINT "YOU WIN!"
170GOTO 10
```




```
GO
LIST
GO
```

Use the PAUSE button to stop the list so you can check it.

```
GO
```



If you would like your computer to keep score, just add these lines. The computer will place them in your program automatically.

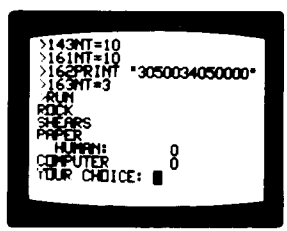
```
H
6H = 0
8C = 0
32PRINT "HUMAN:",H
34PRINT "COMPUTER:",C
145C = C + 1
165H = H + 1
```

If you want to add music, these instructions will do it.

```
141NT = 10
142PRINT "135 x 105 x 10000"
143NT = 2
161NT = 10
162PRINT "3050034050000"
163NT = 2
```

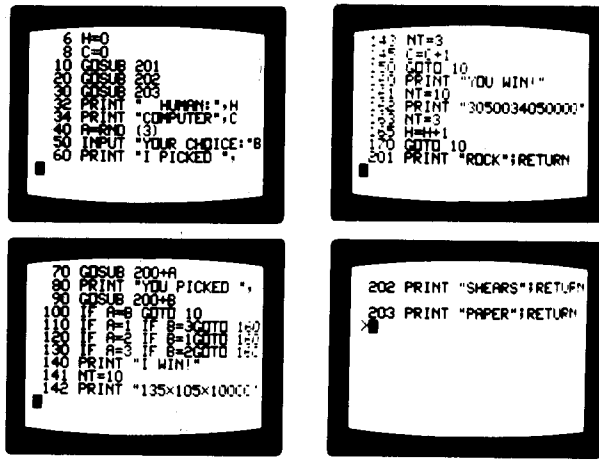
Now RUN your program and see if you can beat your computer.

```
RUN
GO
```



Here is a complete listing of your ROCK, SHEARS AND PAPER game.

```
  1  .ROCK/SHEARS/PAPER
  2  .BY DICK AINSWORTH
 10  H = 0
 20  C = 0
 30  GOSUB 301
 40  GOSUB 302
 50  GOSUB 303
 60  PRINT "    HUMAN:",H
 70  PRINT "COMPUTER:",C
 80  A = RND (3)
 90  INPUT "YOUR CHOICE:"B
100  PRINT "I PICKED ",
110  GOSUB 300 + A
120  PRINT "YOU PICKED ",
130  GOSUB 300 + B
140  IF A = BPRINT "A TIE!";GOTO 30
150  IF A = 1IF B = 3GOTO 240
160  IF A = 2IF B = 1GOTO 240
170  IF A = 3IF B = 2GOTO 240
180  PRINT "I WIN!"
190  NT = 10
200  PRINT "135 x 105 x 10000"
210  NT = 3
220  C = C + 1
230  GOTO 30
240  PRINT "YOU WIN!"
250  NT = 10
260  PRINT "3050034050000"
270  NT = 3
280  H = H + 1
290  GOTO 30
301  PRINT "ROCK";RETURN
302  PRINT "SHEARS";RETURN
303  PRINT "PAPER";RETURN
```



Here is another program that uses subroutines to play each verse. There is no advantage to using subroutines in a small program like this, but it shows how subroutines work. To enter this program into your computer, press RESET to erase the previous program and enter these instructions:

10GOSUB 100
20GOSUB 100
30GOSUB 200
40GOSUB 300
50GOTO 10
100PRINT "1456";RETURN
200PRINT "6 + 665";RETURN
300PRINT "4654";RETURN

When you run this program, you will hear a song with a separate subroutine for each verse. Try typing NT = 15 before running the program to slow the music down.

LESSON 4
ARRAYS

It's often handy to be able to work with a sequence of numbers or letters known as arrays. Arrays can be made up of numbers, letters or musical notes. The numbers stored in an array can, in fact, stand for anything you choose. Arrays are a common method of storing names and addresses, inventory information, and similar data. The advantages of using arrays are that they save memory space and are organized numerically for simple retrieval of stored information.

The concept of an array can be understood by using the Post Office as an analogy. Your Post Office serves many mailboxes, each with a different number,or address. Each mailbox can contain a different message, or, some may contain nothing. Similarly, an array has many addresses, or elements, each of which can contain a separate number. You can command your computer to change the number in a given address of an array, or to fetch the number for use in calculations.

Every character on your keypad occupies one byte of memory when you enter it into a statement. It follows that a number such as −25000 would require six bytes of memory to store as a number, including the minus sign. However, if you store that same number in an array it occupies only two bytes of memory and is easily retrieved by using the array number you stored it in, just as you would use a letter-variable.

Here's how arrays work in your computer. The @ and ✱ characters are your computer's symbols for arrays. There are 1800 bytes of usable programmable memory in Bally BASIC. With every byte of program you store, that number decreases by one byte. After you finish typing in a program, ask the computer to tell you how many bytes of storage you have left over.

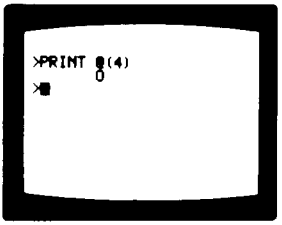
PRINT SZ

GO

The number that prints on the screen will be the remaining memory size, or "SZ." The @ array begins where this leftover memory space starts at the end of the stored program. The first item is at address 0, or @(0). The second item in the array is at address 1, or @(1). The third item is at address 2, or @(2) and so on. As mentioned before, each item in an array occupies two bytes of memory, so if SZ = 100, for example, you have enough room to store SZ ÷ 2, or an array up to 50 items long. Since the @ array begins where the program ends in the memory, it's important to remember that if you modify your program and make it longer, you will cover up the data you had stored in the memory area your program expanded into. To avoid this occurance there is a second array symbol, the ✱ array, which refers to the free memory area in reverse order with ✱(0) starting at the end of the memory and working backwards towards the end of the stored program. The same rules apply as with the @ array with two bytes used per item. So an SZ of 100 would indicate a maximum storage capability of 50 items numbered ✱(0) thru ✱(49). Don't neglect to count the zero as a number when using arrays. It will always be your first useable array address.

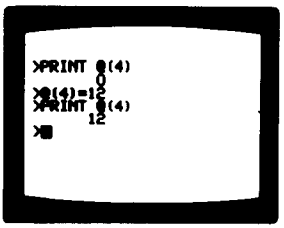
To find the number stored at address 4 in an array, you can tell the computer to print the value of @(4) like this:

RESET
PRINT @(4)
GO



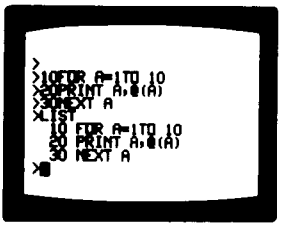
When you RESET the computer, all 1800 memory addresses are cleared and filled with zeroes. So address 4 in the array contains a zero. Store the number 12 at address 4 like this, then check it.

@(4) = 12
GO
PRINT @(4)
GO



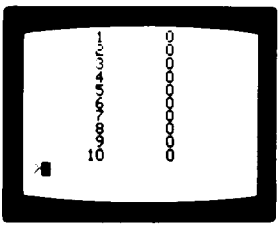
The following program uses a FOR/NEXT loop to list the numbers stored at the first ten addresses in the @ array.

RESET
10FOR A = 0TO 9
20PRINT A,@(A)
30NEXT A
GO
LIST
GO



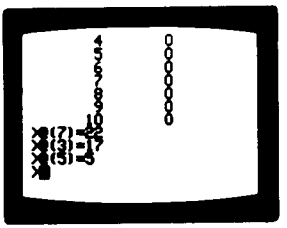
As the variable A advances from 0 to 9, the computer prints 0,and the number stored at address 0, 1 and the number stored at address 1 and so on up to address 9, which contains the last value stored in the array.

RUN
GO



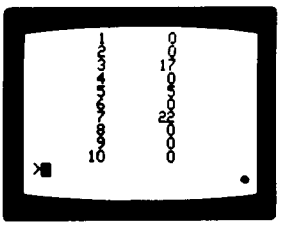
Now enter these instructions. Each time you press GO, the computer follows your instruction, storing number 22 at array address 7, number 17 at array address 3, and number 5 at array address 5.

```
@(7)=22
GO
@(3)=7
GO
@(5)=5
GO
```



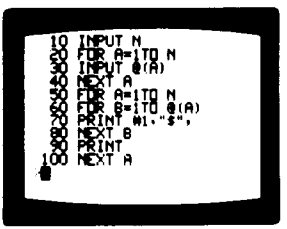
Now run your program and see what numbers are stored at the first 10 array addresses.

```
RUN
GO
```



This next program prints a simple graph, using the array to store the numbers to be plotted.

```
RESET
10INPUT N
20FOR A=1TO N
30INPUT @(A)
40NEXT A
50FOR A=1TO N
60FOR B=1TO @(A)
70PRINT #1,"$",
80NEXT B
90PRINT
100NEXT A
GO
LIST
GO
```



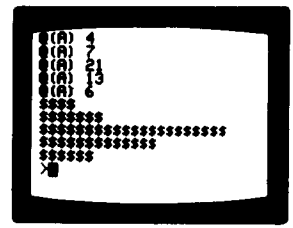
What was the meaning of the # symbol in line 70? It is a formatting symbol that tells your computer to leave a certain number of spaces before printing a character. For a complete explanation see Appendix H under PRINT #A,B.

In line 10, you will set N equal to the number of items in the graph. The loop in lines 20, 30, and 40 stores the value of each item. Lines 50 through 100 are a FOR/NEXT loop using the variable A. This loop prints each item in the graph.

Lines 60, 70 and 80 are a smaller loop that "counts" each item. For example, if @(3) is a 6, this loop will cycle six times and print **$$$$$$** on the screen.

Run your program and draw a bar graph. Enter the number of items, then the value of each item.

RUN
GO
5
GO
4
GO
7
GO
21
GO
13
GO
6
GO



STORING TEXT IN ARRAYS

At the beginning of this lesson you learned that one of the most useful applications of arrays was to store letters and numbers, or, to use a well-known computer expression, "alpha-numeric characters." You know now that your computer doesn't understand words and letters except the keypad command words. To store words in arrays we have to store them as a sequence of numbers that your computer can translate back into letters. Every character and command word on your keypad has a numeric equivalent stored in your computer. These numbers comform to a standard agreed upon by most computer manufacturers called *American Standard Code for Information Interchange,* or ASCII.

To show you which numbers are assigned to which characters we have included an ASCII Character Code Table in APPENDIX A for easy reference. But there is a way to ask your computer to display these codes with two functions built-in to your Bally BASIC.

THE KP AND TV FUNCTIONS

"KP" stands for KEYPAD, and is an expression of the ASCII numerical equivalent of every key on your keypad. "TV" is a function that prints the character the ASCII number stands for when you tell the computer where to look for it, whether in a variable, an array or straight from the keypad when a key is depressed. The following short program will demonstrate how KP and TV can work together to give you ASCII code information about any character.

RESET
10A = KP;PRINT A;RUN

KP works like an input statement because it tells the computer to stop running until it receives a number from you. The difference is that KP takes the first digit, stores it in a variable (A in this program), and begins running the program again without a GO command. The number stored in the variable A is the ASCII code for the key you pressed.

Here is what your computer does:

The program tells the computer to set A equal to the ASCII code for the key you pressed. The next instruction tells the computer to PRINT the number stored in A, and the next instruction tells the computer to RUN, or to go back to the first line in the program, the only line in this case, and begin executing instructions there. GOTO 10 would have done the same thing, but would have used 3 bytes. RUN uses only 1 byte, as do all the other keypad command words. As you may have noticed, this short program has three commands packed into one line. You can do this by using the semicolon to separate commands. This saves memory because each line number uses two bytes and each carriage return (GO) uses one byte. The semicolon uses only one byte and does the same thing. You are limited to 104 bytes per line, including two bytes for the line number and one byte for the carriage return. Remember, when an IF statement is false, all commands following it on the same numbered line are ignored by the computer, which goes on to execute the next line in the program.

To retrieve the information stored in ASCII code and display it as the character it stands for, just tell the computer to set the TV function equal to A. Change the program to print the ASCII code for the key you press plus the character it stands for:

```
10A = KP;TV = A;PRINT A;RUN
```

This program will display the code and the character for any number, letter or computer command word. Then it will wait for your next KP input and do it all over again.

You are now probably saying to yourself, "All I have to do to store text in arrays is to tell the computer to store the ASCII code numbers in consecutive array addresses with the KP function. Then to retrieve the information, I'll just tell the computer to set TV equal to the numbers stored in each address, using a loop that counts up to the number of characters stored."

Okay, if you're ready let's give it a try. Let's use a short loop that will store up to 10 characters or command words as you might use in a word game.

```
RESET
10PRINT "INPUT YOUR MESSAGE";FOR A = 0TO 9
20@(A) = KP
30TV = @(A)
40NEXT A
50CLEAR ;PRINT "TO READ MESSAGE PRESS GO";IF KP#13GOTO 50
60FOR A = 0TO 9
70TV = @(A)
80NEXT A;PRINT ;RUN
```

Now, run the program and put in any 10 letter word. If the word is less than ten letters, fill the rest with spaces. To see your message retrieved and printed on the screen just press GO. After reading your message you can type in a new one because line 80 tells the computer to RUN the program again automatically.

This program uses two FOR/NEXT loops. The first one stores the ASCII codes in consecutive array addresses. The second loop looks into the addresses and sets TV equal to each of them to reprint the message on the screen. Notice the "#" in line 50. It is the sign meaning "not equal to," which told the computer that if you press any key except GO, or ASCII number 13, go back to line 50 and wait for another input.

In the next lesson you will see how arrays can be used to store and play back musical notes.

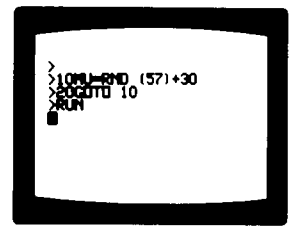
LESSON 5
ELECTRONIC MUSIC

There are four ways you can play music on your computer. The PRINT instruction can be used to play any tune you like. You can also use the MU instruction to convert numbers directly into notes without printing on the screen. The third method involves programming the music synthesizer ports &(16) thru &(23), either directly or through the new Bally BASIC sound synthesizer device variables, and is described on page 114.

The following program sets MU equal to a random number between 31 and 87. Numbers in this range produce musical notes in your TV speaker.

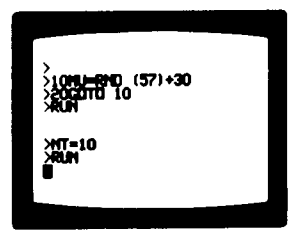
Enter and RUN this random music generator.

RESET
10MU = RND (57) + 30
20GOTO 10
GO
RUN
GO



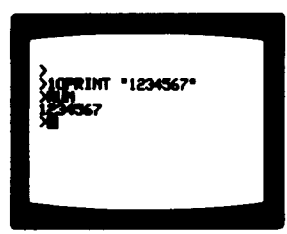
To change the speed of the notes, adjust the built-in note timer controlled by the two-letter variable, NT. HALT your program and set the note time to 10.

H
NT = 10
GO
RUN
GO

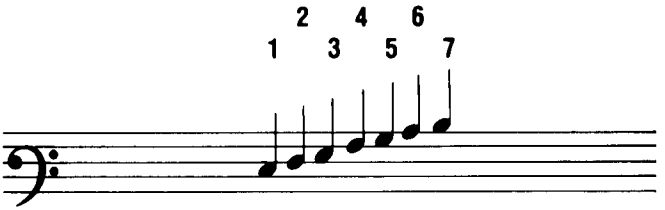


With PRINT and the numbers 1 through 7 you can play a musical scale. The note timer automatically returns to 2 whenever you press RESET.

RESET
10PRINT "1234567"
GO
RUN
GO



Here are the notes you just played:

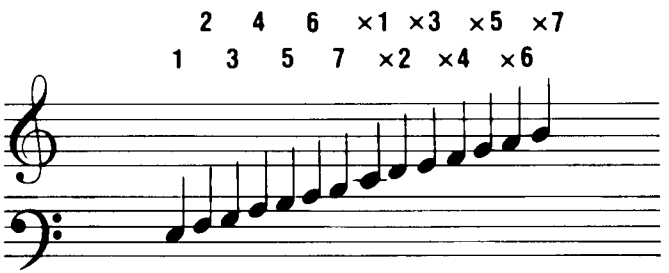


To expand this scale one octave higher, just put a multiplication sign in front of each number.

20PRINT " x 1 x 2 x 3 x 4 x 5 x 6 x 7"
GO
RUN
GO

```
>
>10PRINT "1234567"
>RUN
1234567
>
>20PRINT "x1x2x3x4x5x6x7"
>RUN
1234567
x1x2x3x4x5x6x7
>
```

Your program now plays these notes:

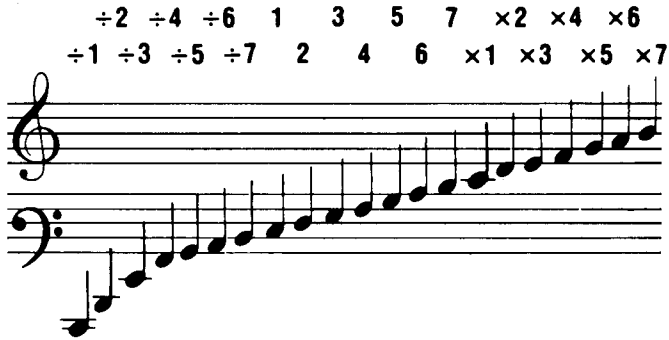


Now add the lowest octave and play your computer's full musical scale. Put the division sign in front of the numbers 1 through 7.

30PRINT " ÷1 ÷2 ÷3 ÷4 ÷5 ÷6 ÷7"
GO
RUN
GO

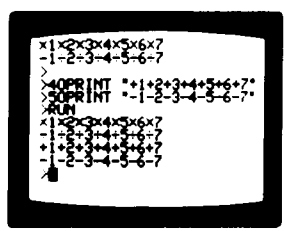
```
>20PRINT "x1x2x3x4x5x6x7"
>RUN
1234567
x1x2x3x4x5x6x7
>
>30PRINT "÷1÷2÷3÷4÷5÷6÷7"
>RUN
1234567
x1x2x3x4x5x6x7
÷1÷2÷3÷4÷5÷6÷7
>
```

Your computer's complete musical scale is now:



Sharps are selected by using an addition (plus) sign in front of the notes, and flats are shown with a subtraction (minus) sign.

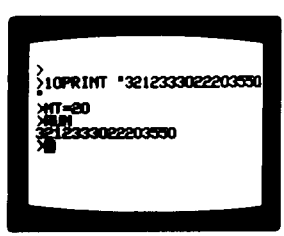
```
40PRINT "+1+2+3+4+5+6+7"
50PRINT "-1-2-3-4-5-6-7"
GO
RUN
GO
```



Always put the sharp or flat sign in front of the octave sign, like this: − +2 or + ×4.

Now RESET the computer and play this tune. Slow the music down by making the note time equal to 20.

```
RESET
10NT=20;PRINT "3212333022203550"
20NT=0
GO
RUN
GO
```

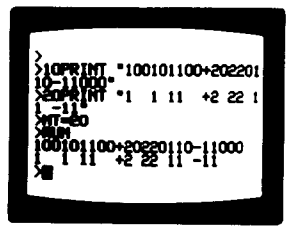


Notice we have set NT equal to zero in the last command in this program. This is to eliminate the long discordant note resulting when the cursor appears as the program stops.

Rhythm can be added two ways. You can space between notes, or add a 0, depending on the sound you want. Try these examples and hear the difference.

We will use the ■ symbol to indicate a SPACE.

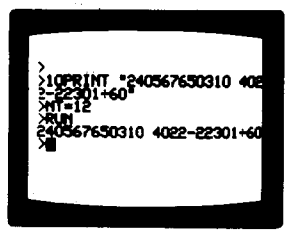
```
RESET
10NT=20;PRINT "100101100+20220110-11000"
20PRINT "1■■1■11■■+2■22■11■-11■■■"
30NT=0
GO
RUN
GO
```



Notice that the notes hold or continue when you use a 0. The space key makes a rest. RUN this program again if you want to listen to the difference.

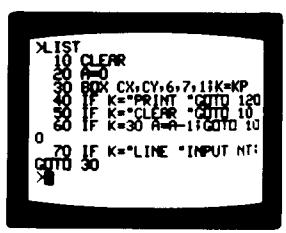
This next program combines everything you have learned. Notice how the space and the 0's set the rhythm.

```
RESET
10NT=12;PRINT "240567650310■4022-22300+60"
20NT=0
GO
RUN
GO
```



Now build a player piano that stores an entire song and then plays it back. You will enter this program in two sections so it will be easier to check.

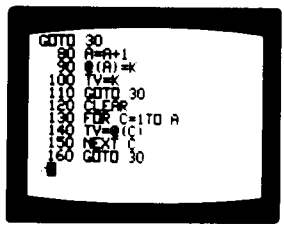
```
RESET
10CLEAR
20A=0
30BOX CX,CY,6,7,1;K=KP
40IF K="PRINT "GOTO 120
50IF K="CLEAR "GOTO 10
60IF K=31A=A-1;GOTO 100
70IF K="LINE "INPUT NT;GOTO 30
GO
LIST
GO
```



NOTE: See Appendix H for explanation of CX and CY functions shown in line 30.

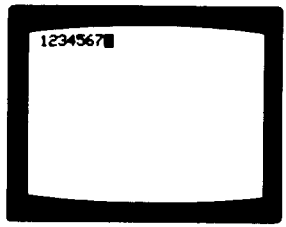
Compare your program with the example, correct any errors, and then enter the second section.

80A = A + 1
90@(A) = K
100TV = K
110GOTO 30
120CLEAR
130FOR C = 1TO A
140TV = @(C)
150NEXT C
160GOTO 30
GO
LIST
GO



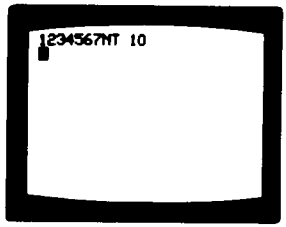
Check your program carefully. When you run it, the screen will go blank. Enter a scale and play it back with the word PRINT.

RUN
GO
1234567
PRINT
PRINT



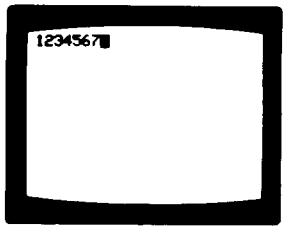
To change the note time, use the word LINE, enter the new note time and press GO.

LINE
10
GO



With this program the GO key is ONLY used after you enter a new note time. Play back at the new note time, using PRINT as before.

PRINT



The word CLEAR is used to clear the memory so you can enter a new song. With ERASE you can back up and change any or all of the notes. Now enter this song. The numbers are shown here in groups of four because there are four beats to a measure. Enter the numbers in a continuous line. Do not press GO at the end of each line.

CLEAR

■ = Space Key

100 + 5	I've been	2002	Can't you
1 + 512	working on the	+ 1232	hear the whistle
3000	rail-	1000	blow-
1000	road.	+ 5000	ing?
4004	All the	4044	Rise up so
1020	live-long	1122	early in the
3000	day.	3000	morn.
000■		000■	
100 + 5	I've been	+ 600 + 7	Can't you
1 + 512	working on the	11 + 71 + 6	hear the captain
3000	rail-	+ 5000	shout-
1033	road. Just to	1000	ing.
3020	pass the	3040	Di-na
2030	time a-	3020	blow your
2000	way.	1000	horn.
000■		000■	
		LINE	
		10	
		GO	
		PRINT	

If you would like to know more about the Player Piano Program, list it and read the following section.

The variable A keeps track of how many notes are stored in the @ array. After clearing the screen and setting A to 0, the computer draws the cursor box and waits for you to enter a number on the keypad. The variable K is set to this number.

Next, the computer checks to see if any words have been entered. If you enter PRINT, the program goes to line 120 to play back the notes. If you enter CLEAR, the computer goes back to the beginning of the program and sets A to 0. Key 31 is the erase key, and if this is pressed, A is reduced by one. The word LINE is used in this program to input a new number for NT, the note time.

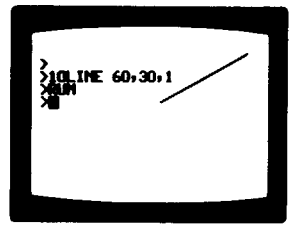
After checking to see if you have entered any special words, the computer adds one to the number stored in A. The new note is added to the @ array (line 90), and is shown on the TV (line 100). GOTO 30 sends the computer back to wait for the next input from the keypad (Line 30).

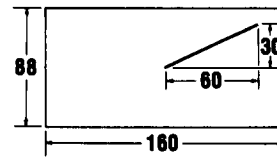
If PRINT is entered, the computer goes to line 120 and starts the playback process. The screen is cleared, and a FOR/NEXT loop is started. Remember that A keeps track of how many notes there are. This part of the program (lines 130, 140 and 150) loops once for each note until all the notes have been written on the TV and played.

LESSON 6
GRAPHICS

With only the words LINE and BOX you can draw an endless variety of pictures, graphs, and designs on your TV. Here's how LINE works.

RESET
10LINE 60,30,1
GO
RUN
GO

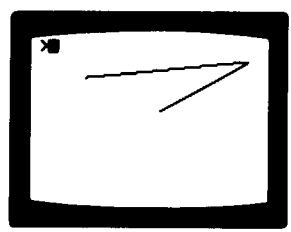


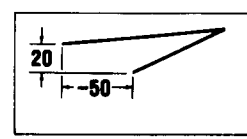


The smallest dot your computer can produce on the screen is called a "pixel," short for "picture element." Your TV screen is 160 pixels wide, (X direction), and 88 pixels high, (Y direction.) Zero is in the center. When you run this program, the computer starts in the center of your screen and draws a line to a point that's 60 pixels to the right of the center (60) and 30 pixels up from the center (30).

After you have run the program, add these instructions to clear the screen and draw the second line.

5CLEAR
20LINE -50,20,1
GO
RUN
GO

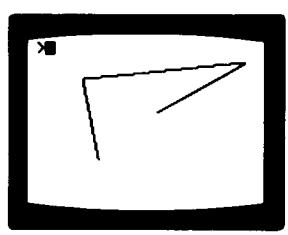


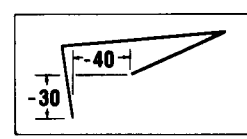


Run the new program and see the computer draw two lines. The computer moved to a point 50 pixels to the left of center (-50) and 20 pixels up from center (20) to draw the second line.

Now add this instruction.

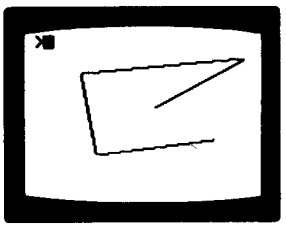
30LINE -40,-30,1
GO
RUN
GO

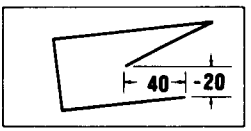




The computer moves to a point that is 40 pixels to the left of center (−40) and 30 pixels down from center (−30). Continue drawing in the lower right section of your screen with this instruction that means 40 to the right (40) and 20 down (−20).

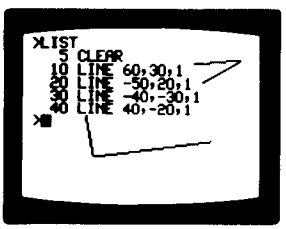
40LINE 40, −20,1
GO
RUN
GO





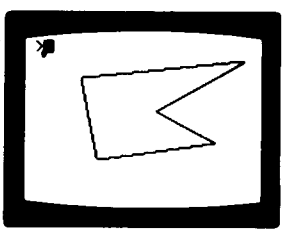
List your program and check to see that you have all the instructions properly entered.

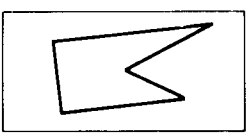
LIST
GO



Finally, draw a line back to the center (0,0) to complete your first graphic design.

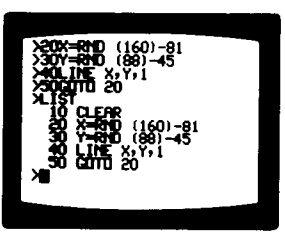
50LINE 0,0,1
GO
RUN
GO





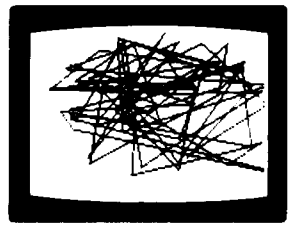
Run this program several times and see how the LINE instruction is used. Erase the program with RESET, and enter the following program that fills the screen with random lines.

RESET
10CLEAR
20X = RND (160) − 81
30Y = RND (88) − 45
40LINE X,Y,1
50GOTO 20
GO
LIST
GO



The computer selects random numbers for X and Y. Then it draws a line to the point on the TV screen that is X pixels right or left of center and Y pixels up or down. It loops back and picks a new X and Y position and then continues drawing.

RUN
GO

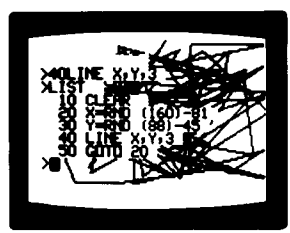


The number 1 after LINE means draw a black line. There are four kinds of lines you can make.

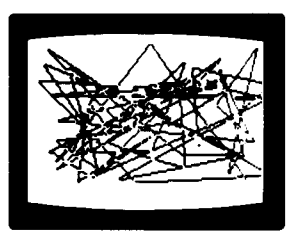
LINE X,Y,1 = Black
LINE X,Y,2 = White
LINE X,Y,3 = Reverse
LINE X,Y,4 = None

Change line 40 and find out what "reverse" lines are.

H
40LINE X,Y,3
GO
LIST
GO

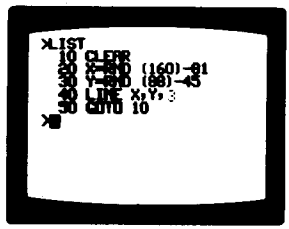


RUN
GO



One trick to use: If you draw anything on the screen with color option 3, (as in LINE X,Y,3) you can erase it simply by drawing it again the same way. Halt your program, clear the screen, and list it.

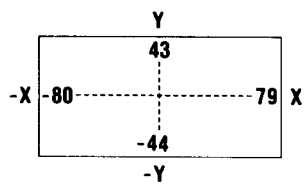
H
CLEAR
GO
LIST
GO



Here's how the computer draws lines that match the size of your TV screen.

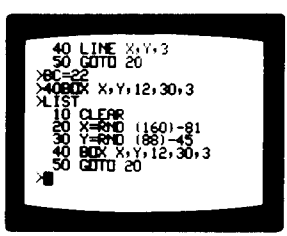
In Line 20, the computer picks a number for X between −80 (on the far left edge of the screen) and 79 (on the right edge of the screen.)

In Line 30, the computer selects a random number for Y that is between −44 (on the bottom edge of your screen) and 43 (on the top edge of your screen.) Bally BASIC will accept line coordinates that go beyond the screen but will print only the portion of the line that falls within the 160x88 pixel area on the screen.



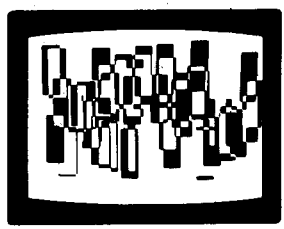
Now change your program and create "reverse" boxes all over your screen. Also change the background color (BC).

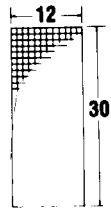
BC = 22
40BOX X,Y,12,30,3
GO
LIST
GO



The random numbers X and Y, position the box on the screen. The next two numbers, 12 and 30, tell the computer how many dots wide and tall to make the box. The last number, 3, reverses as before.

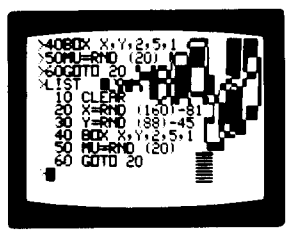
RUN
GO



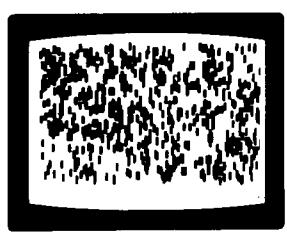


Now make something different. Change the size of the boxes to look like the holes in an IBM card. Change the last number in line 40 to a 1, which will make all the boxes look black. Add some computer music with line 50.

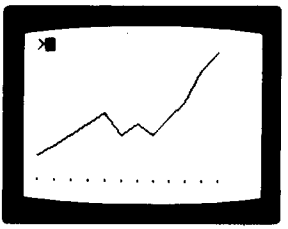
H
40BOX X,Y,2,5,1
50MU = RND (20)
60GOTO 20
GO
LIST
GO



RUN
GO



This next program draws a graph. First, it asks how many numbers you have, then it asks for each number. Finally, it draws a graph that might look like this.



Enter and LIST this part of the program.

```
RESET
10CLEAR
20INPUT "◄–A–►"A
30FOR N=1 TO A
40PRINT N,
50INPUT "?"@(N)
60IF @(N)►87GOTO 40
70NEXT N
GO
LIST
GO
```

```
>60IF @(N)>87 GOTO 40
>70NEXT N
>LIST
  10 CLEAR
  20 INPUT "◄A►"A
  30 FOR N=1TO A
  40 PRINT N,
  50 INPUT "?"@(N)
  60 IF @(N)>87 GOTO 40
  70 NEXT N
>
```

In Line 20, the computer asks how many items the graph will have, then stores the answer in A.

The FOR/NEXT loop prints the number of each item, stores the value in the array @(N), and checks to see if the value is over 87. If it is over 87, it will not fit on the TV screen and the computer goes back to line 40 for a new input.

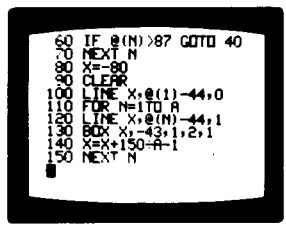
Run this portion of the program.

```
RUN
3
GO
3
GO
21
GO
95
GO
13
```

```
◄A► 3
          1? 3
          2? 21
          3? 95
          3? 13
>
```

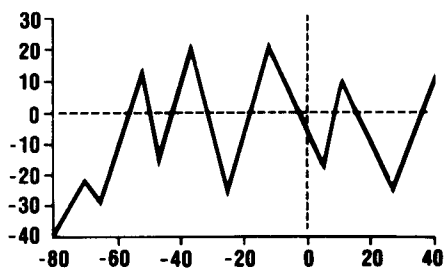
Now add the final section that draws the graph.

```
80X = -80
90CLEAR
100LINE X,@(1)-44,0
110FOR N=1TO A
120LINE X,@(N)-44,1
130BOX X,-43,1,2,1
140X=X+150÷A-1
150NEXT N
GO
LIST
GO
```



To start drawing the graph (line 80), the computer sets X = -80 (the left edge of your screen), clears the screen, and places the starting point for the series of lines that make the graph.

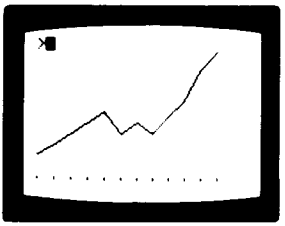
The number @(1) - 44 is the vertical distance (number of pixels) above or below the center of the screen. For example, if the first number in the @ array is 0, then the computer subtracts 44 to place this point on the bottom of the graph.



There are three instructions (120, 130 and 140) in the last FOR/NEXT loop. These instructions are run once for each item in the graph. In line 120, the computer draws a line from the last point to the next point. Line 130 places a small dot at the bottom of the graph. Line 140 changes the variable X to move each point on the graph a short distance to the right. The graph is 150 pixels wide and this distance is divided equally.

Run the program and draw a graph with these twelve figures. Don't forget to push GO after each number.

RUN
GO
12
15
21
28
35
42
28
35
28
39
49
68
79



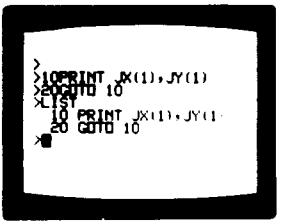
Remember — no single entry can be larger than 87 and no decimal points are accepted.

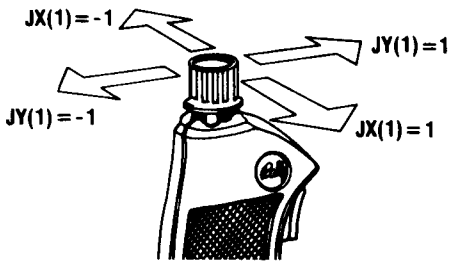
Use your graph drawing program to make a graph of your grocery expenses, your company sales, or your favorite stock.

LESSON 7
VIDEO GAMES

In this lesson you will learn how to build your own video game, using the hand controls. The program will be assembled one piece at a time, so you can see what each section does. First, plug a hand control into the number 1 port (next to the power cord), and then enter this program to see how it works:

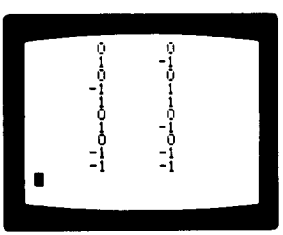
RESET
10PRINT JX(1),JY(1)
20GOTO 10
GO
LIST
GO





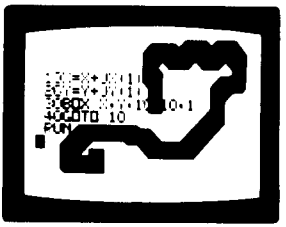
JX and JY are two-letter variables used by your computer to check the position of hand control joysticks 1 through 4. For example: JX (2) = 0 means the joystick on hand control 2 is in the center and has not been pushed either left or right. With the number 1 joystick control centered, JX(1) and JY(1) are zero. Moving the joystick to the right makes JX(1) = + 1, and moving it to the left makes JX(1) = − 1. Similarly, moving the joystick forward or back makes JY(1) either + 1 or − 1. Run the program and change the numbers on your screen by moving the joystick left and right, back and forth. Turning (rotating) the knob has no effect right now.

RUN
GO



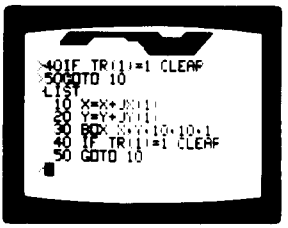
Now use the joystick to move a box on the screen with this program. Two variables, X (left and right) and Y (up and down), keep track of where the box is. When you move the box with the joystick you will be adding +1 or −1 to the variables. Run the program and move the box.

```
RESET
10X=X+JX(1)
20Y=Y+JY(1)
30BOX X,Y,10,10,1
40GOTO 10
GO
RUN
GO
```



The trigger for the first hand control is called TR(1). TR is another two-letter variable that checks the positions of the triggers. TR(1) = 1 when the trigger is pulled and TR(1) = 0 when the trigger is not pulled. Add this line to your program so you can clear the screen by pulling the trigger.

```
H
40IF TR(1)=1CLEAR
50GOTO 10
GO
LIST
GO
```

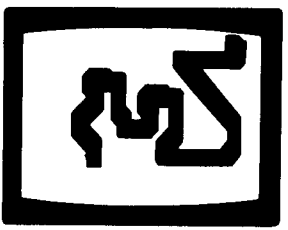


Line 40 checks to see if TR(1) is pulled every time the program cycles. Remember the chapter on IF statements? If line 40 expresses a true statement its value is 1 and the following command, CLEAR, is carried out. If the statement is false its value is 0 and the rest of that line is ignored by the computer, which goes on to the next line. A byte-saving hint--Line 40 could be shortened to read as follows:

```
40IF TR(1)CLEAR
```

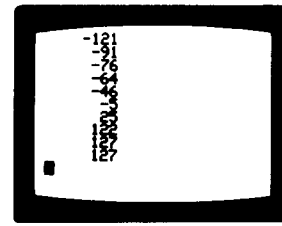
The "equals one" expression isn't needed because the statement has a value of 1 if it's true, and the computer knows this. Now run your program, draw some lines, and clear the screen with the trigger.

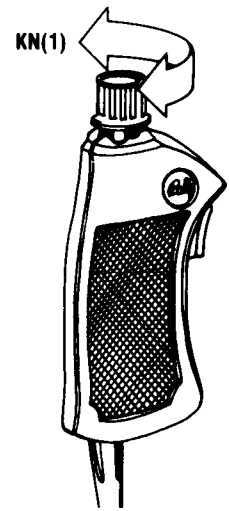
```
RUN
GO
```



Now change your program and see what happens when you turn the knob. The knob on hand control number 1 is called KN(1). Run the program and turn the knob.

RESET
10 PRINT KN(1)
20GOTO 10
GO
RUN
GO

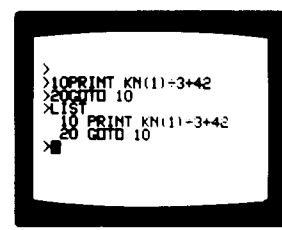




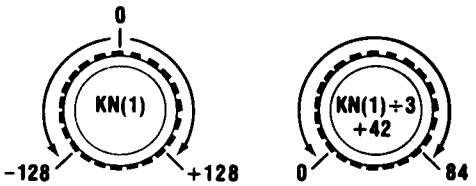
With the knob turned all the way to the left, KN(1) = − 128. With the knob turned to the right KN(1) = 127. Try to dial your age. This is hard to do because the numbers are very close together on the knob.

This next program spreads the numbers out and makes it easier to dial your age.

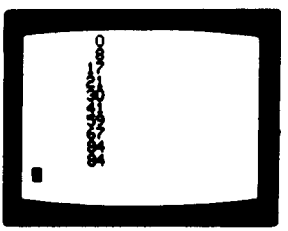
RESET
10PRINT KN(1) + 3 + 42
20GOTO 10
GO
LIST
GO



Here's what you have done to make it easier. KN(1) still has a range from −128 to 127. When you divide KN(1) by 3 this range is reduced to −42 on the left and 42 on the right. When the computer adds 42 to KN(1) ÷ 3, the final range is 0 on the left and 84 on the right. In a similar way you can write an instruction and change the numbers on the dial to match any range you would like. Run this program and see that the knob rotates from 0 to 84.

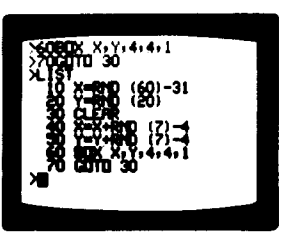


RUN
GO



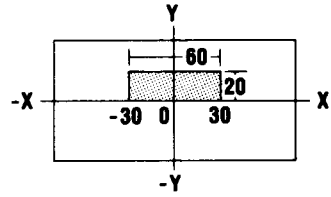
Now you can use the hand control to build your own video game. Begin with this portion of the program that makes a blinking target move around on the screen.

RESET
10X = RND (60) − 31
20Y = RND (20)
30CLEAR
40X = X + RND (7) − 4
50Y = Y + RND (7) − 4
60BOX X,Y,4,4,1
70GOTO 30
GO
LIST
GO



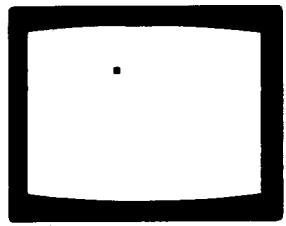
First, the computer picks a value for X between −30 and 30, and then picks a value for Y between 1 and 20. These values for X and Y are in the shaded area of the diagram below. Line 40 and 50 cause the target to wander around the screen. In Line 40, the computer adds a random number to X, moving the target to the right or left. The number added to X is RND (7) − 4. RND (7) is a random number between 1 and 7. Subtracting 4 makes this equal to a random number between −3 and 3.

In Line 50, RND (7) − 4 is added to Y, and this moves the target up or down. The BOX is drawn at X and Y, and the program loops.



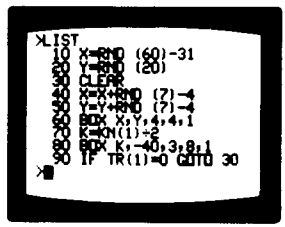
Now run the program and see that it puts a 4 x 4 black box somewhere in the shaded area.

```
RUN
GO
```

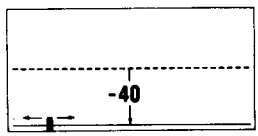


Add a second box at the bottom of the screen. You will move this box left and right with the knob. Notice that you will replace the old line 70 with a new instruction.

```
H
70K = KN(1) + 2
80BOX K, -40,3,8,1
90IF TR(1) = 0GOTO 30
GO
LIST
GO
```

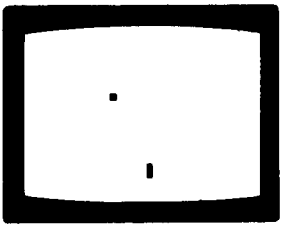


In Line 70, the variable K is set to the value of the knob KN(1) + 2. Line 80 draws a black box that is three squares wide and eight squares tall. The box can be moved left or right as the value of K changes. The center of the box can be moved left or right as the value of K changes. The center of the box will be at −40, near the bottom of your screen.



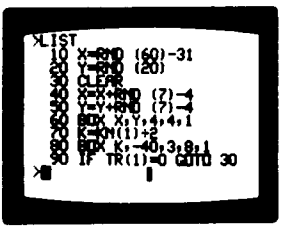
When you pull the trigger, TR(1) = 1. In line 90, the computer goes back to line 30 if the trigger is not pulled and TR(1) = 0. Run the program and see if you can move the second box with the knob.

RUN
GO



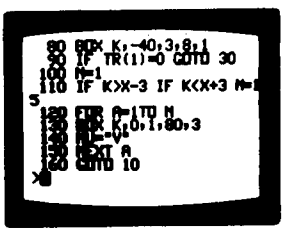
Pull the trigger and see what happens, then list your program.

LIST
GO



When you pulled the trigger, TR(1) = 1. The computer did not go back to Line 30 at the end of your program. Instead, it went on to the next instruction. The following instructions tell the computer what to do when you pull the trigger:

```
100N = 1
110IF K►X - 3 IF K◄X + 3N = 15
120FOR A = 1TO N
130BOX K,0,1,80,3
140MU = "V"
150NEXT A
160GOTO 10
GO
LIST
GO
```



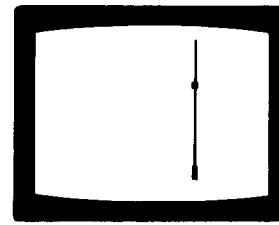
Remember that the variable X moves the target left and right. The "phaser" or box at the bottom of the screen is moved left and right by the variable K. If K = X when you pull the trigger, the phaser and the target are lined up vertically, and the computer will score a hit!

Hitting the target exactly in the center is very hard, so Line 110 allows a near miss to also score. If K is within three pixels of X, then N = 15.

The box in Line 130 is 80 pixels high and only one pixel wide, forming the laser beam. The variable N is set to one in Line 100. If a hit is scored, N = 15. The laser fires N times in the FOR/NEXT loop. For a miss, the beam fires once, and for a hit it fires 15 times. The MU instruction plays music without printing the notes on the screen.

After each shot, the program loops back to the very beginning and continues to move the target to a new random position until you press the trigger. Now run the program and try your luck.

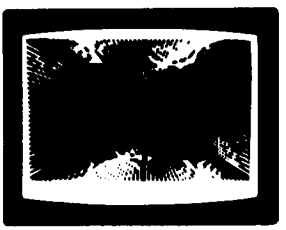
RUN
GO



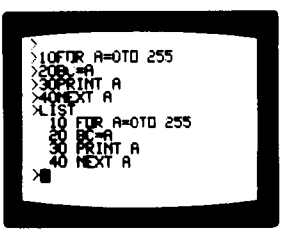
The program could be a two-player game if you use another hand control instead of the computer to move the target. The inputs for the number two hand control are JX(2), JY(2), KN(2), and TR(2). Also, if you have optional hand controls to plug into ports 3 and 4, you can program them in a similar fashion, with TR(3) and TR(4), etc., being the proper designations. You could also improve this program by keeping track of and printing the score, coloring the screen to show a hit, reversing the black and white for night effects, and many other variations. Try doing these by yourself!

LESSON 8
VIDEO ART

In this lesson you will learn how to use the power of your computer to create interesting and beautiful designs. Begin with this program that shows you all the colors in your computer and prints each color number.

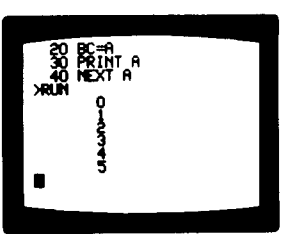


RESET
10FOR A=0TO 255
20BC=A
30PRINT A
40NEXT A
GO
LIST
GO



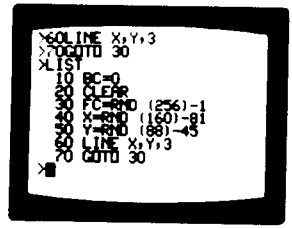
The background color is controlled by the two-letter variable (BC) and can be any number you select from 0 to 255. When you press RESET the computer sets BC = 7 (white) and the foreground color, FC = 0 (black). In this program, the computer begins with color number 0 (black) and shows each color and its number. Now run your program and see all the colors you can select from.

RUN
GO



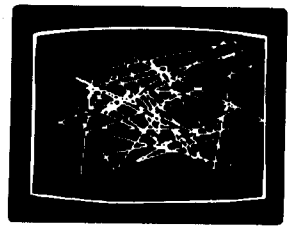
Enter this program and let the computer select the color while drawing random lines on your screen.

```
RESET
10BC=0
20CLEAR
30FC=RND(256)-1
40X=RND(160)-81
50Y=RND(88)-45
60LINE X,Y,3
70GOTO 30
GO
LIST
GO
```



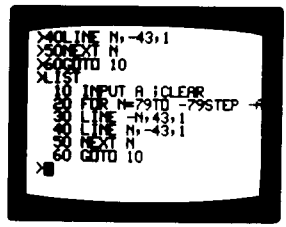
First, the computer sets the background color (BC) to black and clears the screen. In Line 30, the foreground color (FC) is picked at random from the 256 choices (0 to 255). The minus sign is to insure that 0 is included in the choice of numbers, because RND (256) means from 1 to 255. Then the computer draws a random line and goes back to instruction 30 to pick a new color and draw the next line.

```
RUN
GO
```



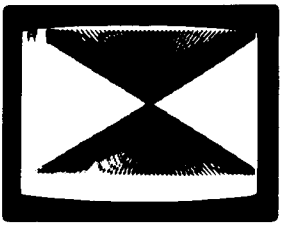
Use the computer to draw a pattern of lines with this program. You will add colors later.

```
RESET
10INPUT A;CLEAR
20FOR N=79TO -79STEP -A
30LINE -N,43,1
40LINE N,-43,1
50NEXT N
60GOTO 10
GO
LIST
GO
```



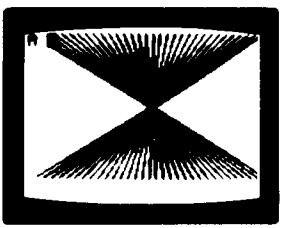
The computer will ask you to input a value for A. This adjusts the spacing between the diagonal lines. Try a spacing of 3 for a start.

RUN
GO
3
GO



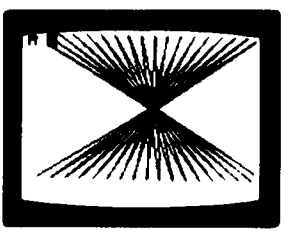
The computer is asking for a new value for A. Try a spacing of 5.

5
GO



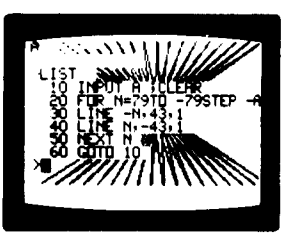
Now try a spacing of 9.

9
GO



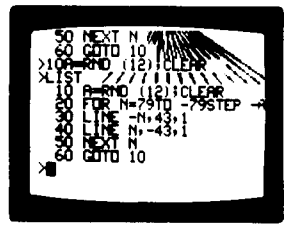
By just changing one number, you have created three different designs. Now let the computer select the spacing. You must halt the program before you can change it.

H
LIST
GO



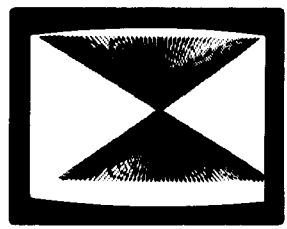
Now make the spacing random with this new instruction.

```
10A = RND (12);CLEAR
GO
LIST
GO
```



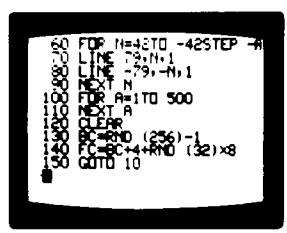
Run your program and let your computer change the design.

```
RUN
GO
```



Complete your design and color it with these additional instructions.

```
H
60FOR N = 42TO  - 42STEP  - A
70LINE 79,N,1
80LINE  - 79,  - N,1
90NEXT N
100FOR A = 1TO 500
110NEXT A
120CLEAR
130BC = RND (256) - 1
140FC = BC + 4 + RND (32) x 8
150GOTO 10
GO
LIST
GO
```

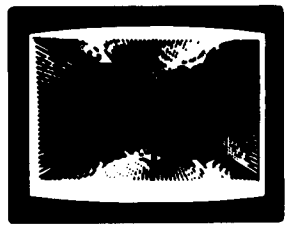


Lines 60, 70, 80 and 90 draw the second half of the design.

A slight pause is added in lines 100 and 110, letting you see the pattern clearly before it changes again.

The background color is selected at random in Line 130, and in the next line the foreground color is adjusted to contrast with the background color.

RUN
GO



COLOR WHEEL

Here is a color wheel you will use often because it helps you select colors and their numbers. Moving the number on hand control left and right selects the color. Moving it forward and backward selects the intensity. Pulling the trigger gives you a printout on the screen that shows that particular color number (0 to 31) color intensity (0 to 7) and the computer number (0 to 255).These numbers refer to the background color only. The foreground color is adjusted automatically so that you can read the numbers.

```
RESET
10C=C+JX(1)
20IF C►31C=31
30IF C◄0C=0
40I=I+JY(1)
50IF I►7I=7
60IF I◄0I=0
70BC=C*8+I
80FC=BC+12
90IF TR(1)=0GOTO 10
100PRINT C,I,C*8+I
110GOTO 10
GO
LIST
GO
```

This program uses two variables, C and I, to keep track of the color number and the intensity number. Both are adjusted by the hand control. JX(1) controls color and JY(1) controls intensity.

Lines 20 and 30 keep C between 0 and 31. Lines 50 and 60 keep I between 0 and 7.

The background color is set to the color number times eight plus the intensity number.

If the trigger is not pulled, the program loops back to line 10. Pulling the trigger prints the numbers in line 100 before looping back to line 10.

LESSON 9
THREE VOICE MUSIC WITH BALLY BASIC

By George Moses

The sound synthesizer in your Bally Arcade has 3 voices that you can independently set to any musical frequency. The single note music you've seen described in this book and the tones you hear whenever you press the keypad are produced by voice A. To hear voices A, B and C at the same time you must first set Note Time **NT = 0** to release voice A from control of the keypad one-note music system. Then set the Master Oscillator **MO = 49** because all 3 voices reference this frequency each time they pulse.

All that remains is to set the three voices to full volume. Set the sound synthesizer volume variables to these values: **VA = 15;VB = 15;VC = 15**. Now all three voices are ready to receive their tone frequency commands. Let's play a chord! Type in **TA = 67;TB = 53;TC = 44**. What you should be hearing now is a chord using middle-C **(TA)** accompanied by E **(TB)** and G **(TC)**. To turn off the chord either use the RESET button or downward pointing arrow symbol which sets all sound variables to zero. Keep in mind, whenever you RESET your computer all sound registers are set to zero except MO, which the computer sets to 71, VA is reset to 15 and NT becomes 2. So to regain control of all three voices you have to reenter the values previously described.

When Jay Fenton wrote Bally BASIC he left us total access to each memory location through the peek and poke functions. This is the method used here for storing our musical tone data and retrieving it so the computer can play it. Each byte of memory in the Arcade has a specific address. The text area of memory, which we use for programming begins at address −24576 and ends at −22777 which adds up to 1800 bytes of usable memory. Line numbers 1 through 12 in the program are called REM (.) statements. Whenever a period follows a statement number the computer ignores everything following the period in that statement. But the numbers after the period take up space and reserve a vast area of memory at the beginning of the text area for poking data into.

Begin by putting in REM statements 1 through 12 with a period followed by 97 bytes of space-occupying numbers in each statement. Put in the rest of the program as you see it. No spaces are needed anywhere except between words that you will be reading on the screen for instruction, so don't waste memory by using unnecessary spaces. Before running the program save it on tape for later use. With the tape player on RECORD and with the plug in your MIC socket type in **:PRINT GO**. When the cursor reappears the load is completed (about 20 seconds).

Using the Music Program

To start the program use the command **GOTO 50**. Using **RUN** sometimes causes the computer to crash when it tries to execute the empty REM space at the front end of the text area of memory. After the command **GOTO 50** you'll see the number −24573 on the screen. That's the memory address of the first chord the computer is waiting for you to input.

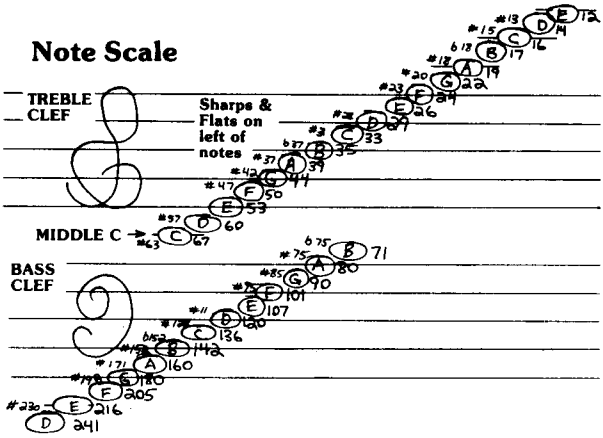


FIG. 1

Using the music chart (FIG. 1) input the proper numbers for the three voices in the first chord. Next, you will be asked for the **DURATION** of the chord. The duration of all notes is determined by the shortest note in the entire song which will have a duration of 1. If that note is a 16th note then all 16th notes = 1, 8th notes = 2, ¼ notes = 4, ½ notes = 8 and whole notes = 16. The duration of any chord is equal to the shortest note in that chord. If you have a note of longer duration than others in the same chord you carry it into the next chord for the remainder of its duration by inputting its tone value into the same voice again. Anytime you set a voice to the same frequency for two or more consecutive chords it will sound as one continuous, but longer note. If, however, you want to sound the same tone as two distinct notes in a row you will have to input it into a different voice for the second beat. To illustrate, let's put in the first four measures of "Chopsticks", a familiar tune (FIG. 2).

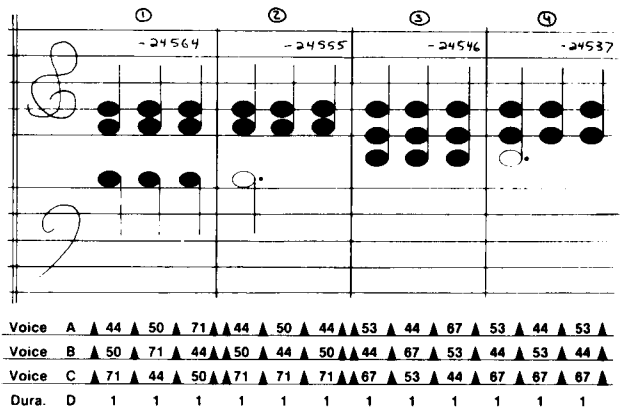


FIG. 2

The voices, you will notice, are constantly being switched to repeat the same chords in consecutive staccato beats. Notice measures two and four each have a half note in voice C. A half note is twice as long as a quarter note so it would normally be carried for two chords. But since these are dotted half notes (a dot extends a note half again as long) they are carried through three chords in voice C. FIG. 2 shows how you should mark your sheet music, indicating the note values for voices A, B, C and then the duration, D. Input each column A through D, then move to the next column and repeat. Stop at the end of each measure and write the memory address currently displayed on the screen. This will allow you to return easily to that location if you want to listen to or correct music beginning with the chord following that number.

Listening to Your Music

A good time to proof your music for errors is at the end of every measure. When you've written the last address at the end of a measure, input the number 333. (Any number larger than 256 will work). The computer will ask you to input the **STARTING ADDRESS**. To start at the very beginning of the song just input **B** and press **GO**. To start at the beginning of any measure, input the address you wrote at the end of the previous measure and press **GO**. The song will play up to the last chord you have input. Then you will be asked to make a choice: [1] REPLAY [2] INPUT [3] CHANGE. Push [1] and you'll hear the same music play again. Push [2] and the screen will clear and display the memory address you stopped at. Your computer is now ready to accept more musical data. Push [3] if you want to change or correct previous inputs. The computer will ask you to input ADDRESS OF CHORD. Again, inputting B will start you at the beginning of the song, address −24573. Or you may start at the beginning of any measure by inputting the address written at the end of the previous measure. When you go back and correct a chord the computer modifies the byte following the one you're poking into. This means you have to correct each consecutive memory address until you reach the location at which you last stopped. This is only a minor inconvenience if you proof each measure as you finish inputting it.

Saving Your Music on Tape

This is a simple matter of using the **:PRINT** command with your tape recorder on RECORD and your cord from the BASIC cartridge connected to the MIC socket. However you may want to delete the lines you don't need first to avoid accidentally starting the program on line 50 and modifying your hard-earned data. Once your song is complete there are only 3 lines of the program needed to play it. Line 120 initializes your sound synthesizer. Line 130 plays your song from memory locations A – 1 through E – 2. Line 140 refers to line 130 as a subroutine and shuts off the sound ports when the song ends with the downward arrow symbol. In fact, line 140 should be shortened to read:

140 GOSUB 130;↓ Or, to hear your song play twice, as in multi-stanza songs, change it to read:

140 GOSUB 130;GOSUB 130;↓

You can play the entire song as many times as you write GOSUB 130 in line 140. And if you want to repeat parts of the song as in the repeat sections of your sheet music you simply type in the new values of A (beginning of section) and E (end of section) before the GOSUB 130. For example:

140 GOSUB 130;A = – nnnnn;E = – nnnnn;GOSUB 130;↓ will play the entire song once then reset the values of A and E and play the data between those locations. Finally, the downward arrow symbol will shut off the sound.

A Few Final Touches

To make your music play at just the right speed, we've provided a variable T in line 110. The larger the value of T the slower your music will play, so adjust it accordingly. Line 110 would be the perfect place for you to print the title of your song on the screen like this:

110 T = 50;CY = 0;PRINT "■■■■■■■■CHOPSTICKS"

Use 8 spaces after the quotation marks to center the 10 letter name on the 26 character line and the CY = 0 command to center the line vertically on the screen.

Line 100 changes the FC and BC each time you begin your song. This line isn't necessary but gives your screen a nice appearance if you have the memory space. The rest of the statements in the program are there to make it easier for the user to input data, but can be eliminated before storing the song on tape. In fact, if your song is extremely long and you run out of REM storage space you can take out lines 145 through 180, change 140 to **140 GOSUB 130;** and add as many bytes of additional REM storage as you have memory space for. **PRINT SZ** will give you the exact number of bytes you have left. Follow these directions and you should have many happy music-filled hours with your Bally Arcade!

THREE VOICE MUSIC PROGRAM

```
 1  .123456789012345678901234567890123456789012345678901234567890123456789
    012345678901234567890123456789012345678901234567
 2  .123456789012345678901234567890123456789012345678901234567890123456789
    012345678901234567890123456789012345678901234567
 3  .123456789012345678901234567890123456789012345678901234567890123456789
    012345678901234567890123456789012345678901234567
 4  .123456789012345678901234567890123456789012345678901234567890123456789
    012345678901234567890123456789012345678901234567
 5  .123456789012345678901234567890123456789012345678901234567890123456789
    012345678901234567890123456789012345678901234567
 6  .123456789012345678901234567890123456789012345678901234567890123456789
    012345678901234567890123456789012345678901234567
 7  .123456789012345678901234567890123456789012345678901234567890123456789
    012345678901234567890123456789012345678901234567
 8  .123456789012345678901234567890123456789012345678901234567890123456789
    012345678901234567890123456789012345678901234567
 9  .123456789012345678901234567890123456789012345678901234567890123456789
    012345678901234567890123456789012345678901234567
10  .123456789012345678901234567890123456789012345678901234567890123456789
    012345678901234567890123456789012345678901234567
11  .123456789012345678901234567890123456789012345678901234567890123456789
    012345678901234567890123456789012345678901234567
12  .123456789012345678901234567890123456789012345678901234567890123456789
    012345678901234567890123456789012345678901234567
50  CLEAR ;NT=0;B=-24573;IF E=0E=B
55  FOR N=0TO 3;IF N=3NT=1;INPUT "DURATION"D;NT=0;GOTO 90
60  PRINT #1,E,;INPUT " "J;IF J►256INPUT "STARTING AD
    DRESS?■"A;GOTO 100
70  J=J-127;IF J◄0J=J+255
80  @(N)=J;NEXT N
90  FOR N=1TO D;FOR A=0TO 2;%(E+A)=%(E+A)+256*256+@(A);NEXT
    A;E=E+3;IF E=-23364%(E)=0
95  NEXT N;GOTO 55
100 CLEAR ;BC=RND (32)*8;FC=RND (32)*8-3;NT=0
110 T=50
120 MO=49;VA=12;VB=12;VC=12;GOTO 140
130 FOR C=A-1TO E-2STEP
    3;TA=%(C)+256+127;TB=%(C+1)+256+127;TC=
    %(C+2)+256+127;FOR D=1TO T;NEXT D;NEXT C;RETURN
140 GOSUB 130;↓;CY=0;PRINT "■[1]■REPLAY";PRINT "■[2]■INPUT ";PRINT
    "■[3]■CHANGE
145 R=KP;IF R=49GOTO 100
150 IF R=50GOTO R
160 IF R=51INPUT "■ADDRESS OF CHORD?"E;GOTO 50
170 GOTO 145
```

PROGRAMS

Here is an assortment of programs you can enter and run immediately. Pick a short program to begin with. If you have any difficulty, return to the Introduction Section, page 17, for assistance.

If you make a mistake in punctuation, (as in leaving out a comma), the computer cannot run your instruction. If this happens the computer will print the instruction on the screen with a question mark in the position of your error, to show you where your mistake is.

If you are using a program designed for one player, be sure to use hand control number one. Programs for two players use hand controls one and two only.

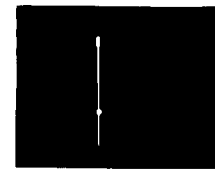
If at any time you wish to see your program, press WORDS LIST GO and your computer will show you all the lines you have entered.

You can modify these programs any way you like. Change or add to the instructions and make the computer do something different. When you add instructions to your program, number the new line to fit between the existing lines. For example, if you want to add an instruction after line 30 and before line 40, number your instruction line 33 (or any number between 31 and 39).

COMPUTER GAMES

PHASER PHUN

Try your skill as the computer moves the target. The first player's knob moves the phaser left or right and the trigger shoots.



```
  1 .PHASER PHUN
  2 .BY DICK AINSWORTH
 10 X=RND(60)-31
 20 Y=RND(20)
 30 CLEAR
 40 X=X+RND(7)-4
 50 Y=Y+RND(7)-4
 60 BOX X,Y,4,4,3
 70 K=KN(1)+2
 80 BOX K,-40,3,8,1
 90 IF TR(1)=0GOTO 30
100 N=1
110 IF K►X-3IF K◄X+3N=15
120 FOR A=1TO N
130 BOX K,0,1,80,3
140 MU="4"

150 BC=A*8
160 NEXT A
170 FC=7
180 BC=8
190 GOTO 10
```

You can make this a two-player game by changing these lines.

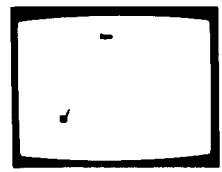
```
40 X=X+JX(2)*3
50 Y=Y+JY(2)*3
```

Player two controls the target while player one shoots.

ANTI-AIRCRAFT GUN

Player one moves the gun with the knob and shoots with the trigger.

Player two moves the plane right or left with JX(2) and controls the speed with the knob.

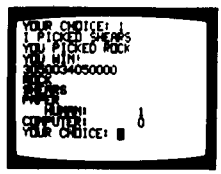


```
 1 .ANTI-AIRCRAFT GUN
 2 .BY BOB OGDON
10 CLEAR ;W = -75;V = 30;C = 0
20 BC = 22;FC = 0
30 BOX -51,-30,5,5,1
40 D = KN(2)
50 IF D►50S = 15;GOTO 80
60 IF D►-50S = 10;GOTO 80
70 S = 5
80 W = W + JX(2) x S
90 IF W►70GOTO 330
100 BOX 0,6,160,58,2
110 BOX W-4,34,2,1,1
120 BOX W,32,10,3,1
130 BOX W+5,32,1,1,1
140 IF TR(1)GOTO260
150 P = KN(1)
160 IF P◄-120X = -46;Y = -23;GOTO 190
170 IF P◄120X = -44;Y = -24;GOTO 190
180 X = -43;Y = -25
190 LINE -48,-27,0
200 BOX -43,-23,10,10,2
210 LINE X,Y,1
220 IF X = -46U = -25
230 IF X = -44U = 7
240 IF X = -43U = 32
250 IF TR(1) = 0GOTO 40
260 IF U = CGOTO 40
270 C = U
280 LINE U,V,1
290 NT = 7;MU = "V"
300 IF U-W◄5IF U-W►-6GOTO 320
310 GOTO 40
320 GOSUB 400;GOTO 350
330 CX = -50;CY = 0
340 PRINT "TOO BAD YOU MISSED"
350 IF TR(2)GOTO 10
360 GOTO 350
400 FOR Z = 30TO -20STEP -20
410 BOX W+2,Z+4,1,2,1
420 BOX W,Z,3,10,1
430 BOX W,Z-6,1,1,1
440 BOX 0,6,160,58,2
450 NEXT Z
460 BC = 74
470 FOR N = -5TO 5
480 LINE W,-25,0
490 LINE N x RND (5) + W,-25 + RND (10),3
500 MU = 1
510 NEXT N
520 RETURN
```

ROCK/SHEARS/PAPER

Enter 1, 2, or 3 and press GO to select Rock,Shears, or Paper. The computer will also make a guess, and then score the results. Here are the rules:

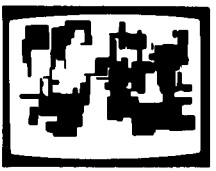
ROCK breaks SHEARS
SHEARS cut PAPER
PAPER wraps ROCK



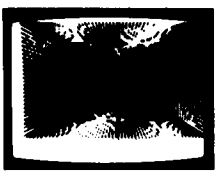

```
  1  .ROCK/SHEARS/PAPER
  2  .BY DICK AINSWORTH
 10  H=0
 20  C=0
 30  GOSUB 301
 40  GOSUB 302
 50  GOSUB 303
 60  PRINT "    HUMAN:",H
 70  PRINT "COMPUTER:",C
 80  A=RND(3)
 90  INPUT "YOUR CHOICE:"B
100  PRINT "I PICKED ",
110  GOSUB 300+A
120  PRINT "YOU PICKED ",
130  GOSUB 300+B
140  IF A=BPRINT "A TIE!";GOTO 30
150  IF A=1IF B=3GOTO 240
160  IF A=2IF B=1GOTO 240
170  IF A=3IF B=2GOTO 240
180  PRINT "I WIN!"
190  NT=10
200  PRINT "135 x 105 x 10000"
210  NT=3
220  C=C+1
230  GOTO 30
240  PRINT "YOU WIN!"
250  NT=10
260  PRINT "3050034050000"
270  NT=3
280  H=H+1
290  GOTO 30
301  PRINT "ROCK";RETURN
302  PRINT "SHEARS";RETURN
303  PRINT "PAPER";RETURN
```

COLOR WAR

One player tries to fill the screen with colored boxes, while the other tries to erase the pattern. The triggers are the secret. If your trigger is in the same position as your opponent's, the screen fills. If your trigger is in the opposite position, the pattern begins erasing itself. Playing this game is like controlling a light with two switches, with one player trying to turn the light on and the other trying to turn it off. Instead of a light, this program creates a pattern that either fills or erases. The two knobs control the colors of the pattern and background.



```
  1  .COLOR WAR
  2  .DICK AINSWORTH
 10  CLEAR
 20  BC = KN(1) + 5 x 5
 30  FC = KN(2) + 5 x 5
 40  X = RND (140) - 80
 50  Y = RND (70) - 35
 60  A = RND (25)
 70  B = RND (25)
 80  IF TR(1) = TR(2)C = 1
 90  IF TR(1)#TR(2)C = 2
100  BOX X,Y,A,B,C
110  GOTO 20
```



```
  1  .SPIRAL 1
  2  .BY DICK AINSWORTH
 10  S = RND (10);L = 1;M = 1
 20  FOR N = 79TO -79STEP -S
 30  LINE -N,43,L
 40  LINE N,-43,M
 50  NEXT N
 60  FOR N = 42TO -42STEP -S
 70  LINE 79,N,L
 80  LINE -79,-N,M
 90  NEXT N
100  FOR A = 1TO 500
110  NEXT A
120  CLEAR
130  BC = RND (256)
140  FC = BC + 4 + RND (32) x 8
150  GOTO 10
```

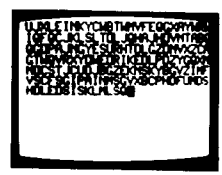
ELECTRONIC MUSIC

COMPOSITION IN A

Enter the notes you wish to hear and then enter PRINT. The computer will play the first note; the first and second; the first, second, and third; and so on until it plays all the notes.

```
 1.COMPOSITION IN A
10  A=1
20  NT=5
30  K=KP
40  IF K=116GOTO 90
50  TV=K
60  @(A)=K
70  A=A+1
80  GOTO 30
90  CLEAR
100 FOR N=1TO A-1
110 FOR P=1TO N
120 TV=@(P)
130 NEXT P
140 CLEAR
150 NEXT N
160 GOTO 20
```

```
 1  .COMPOSITION A TO Z
10  CLEAR
20  A=RND (26)+64
30  MU=A
40  TV=A
50  GOTO 20
```



COMPOSITION IN F

Just enter the total number of notes and press GO . The computer will write and play a composition. Enter the number 15, for a start. The computer takes a while to work out the details, so you will have a short wait. Longer compositions can take several minutes to prepare.

```
  1 .COMPOSITION IN F
 10 CLEAR ;B=4;C=B x B
 20 FOR D=1TO B
 30 @(D)=0
 40 NEXT D
 50 D=C+1
 60 E=6
 70 INPUT F
 80 FOR G=DTO D+F-1
 90 H=A
100 A=A+1
110 I=A
120 J=C
130 K=0
140 FOR L=1TO B
150 J=J+2
160 M=H+J
170 N=I+J
180 IF M#0H=H-J
190 IF N#0I=I-J
200 IF M=NGOTO 220
210 @(L)=RND (E)
220 K=K+@(L)
230 NEXT L
240 @(G)=K
250 IF A=C-1A=0
260 NEXT G
270 INPUT NT
280 CLEAR
290 FOR L=DTO D+F-1
300 TV=@(L)+"A"-B
310 NEXT L
320 NT=2
```

PLAYER PIANO

```
  1 .PLAYER PIANO
  2 .BY JAY FENTON
 10 CLEAR
 20 A=0
 30 K=KP
 40 IF K="PRINT"GOTO 120
 50 IF K="CLEAR"GOTO 10
 60 IF K=31A=A-1;GOTO 100
 70 IF K="LINE"INPUT NT;GOTO 30
 80 A=A+1
 90 @(A)=K
100 TV=K
110 GOTO 30
120 CLEAR
130 FOR C=1TO A
140 TV=@(C)
150 NEXT C
160 GOTO 30
```

See the electronic music section for complete details. Your controls for this program are:

PRINT to play the notes you entered.
ERASE to back up and remove notes from the screen.
LINE to enter a new note time (Press GO after you enter the number).
CLEAR to clear the notes from memory so that you can enter new music to be played.

BAGPIPES

NOTE: Input the PLAYER PIANO program. Next, RUN the program and input these numbers. Put in the numbers from the left column, then the next column to the right.

405654	46 x 2x164
−70 x 2 x 106	606605
406654	46 x 2 x 164
502300	505505
405654	46 x 2 x 164
−70 x 2 x 106	60 x 1 x 20 x 3
x 406654	x 4 x 2 x 1654
504401	605400

MELODY

506
70 × 1
× 400
× 300
× 300
× 200
600
000
70 × 1
+ × 10 × 2
× 700

× 600
× 500
× 500
× 500
× 5 × 4 × 2
76 − 6
506
70 × 1
× 400
× 300
× 300

× 200
600
70 × 5
× 406
× 300
× 200
× 10 × 2
× 10 × 2
× 100
0

MARCH

5000
+ 400 + 4
5000
034 + 4
50 × 10
50006
7000
0223
4000
3003
4000

0223
4070
600 − 6
5000
034 + 4
5000
+ 400 + 4
5000
+ 034 + 4
50 × 10
× 200 × 1

× 1000
01416
× 1000
7006
5000
05 × 1 × 4
× 3000
× 300 × 2
× 1000
0

MARINE'S HYMN

13
5050
5050
500 × 1
5034
5050
4200
1000
0013
5050
5050
500 × 1

5034
5050
4200
1000
00 × 17
6040
6040
5006
50 × 17
6040
6 × 100

5000
0013
5050
5050
500 × 1
5034
5000
5000
6000
7000
× 1000

GOLDEN SLIPPERS

■ = REST (USE SPACE KEY)

45
60606545
60606■45
6060656 – 7
60505■34
5050534
50505■34
50 – 7■6050
4000000■
10000■40
6050410■

20000■50
– 7060520■
30303040
50000■30
40304050
6000000■
10000■40
6050410■
20000■50
– 7060520■
30303040

50000■ – 70
6■6■5■5■
4000000■
40000■ – 70
× 20 × 10 – 740■
50000■ × 10
– × 30 × 20 × 150
606060 – 70
× 10000■60
– 7060 – 70 × 10
× 2000000■

40000■ – 70
× 20 × 10 – 740■
50000■ × 10
– × 30 × 20 × 150■
606060 – 70
× 10000■ – × 30
× 20 × 20 × 10 × 10
– 70000000

PLAYER PIANO
STARS AND STRIPES FOREVER

5000
5043
30 + 23
3000
00 + 23
30 + 23
5035
4000
2002
20 + 12
20 + 12

4000
0032
3500
6060
2000
00 × 50
× 50 × 4 × 3
× 30 + × 2 × 3
× 3000
00 + × 2 × 3
× 30 + × 2 × 3

× 4 × 3 × 27
× 2000
× 10 × 10
× 107 × 1
– × 30 × 2 × 1
× U000
0 × 1 × 2 × 3
× 5 × 1 × 2 × 3
× 556 × 3
× 2000
× 1

COMPOSITION IN L

With this program you can create your own electronic music. After you select a length for the song and enter a series of notes, the computer will play the composition.

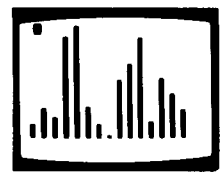
Run the program and enter (10) as a value for L. After you press GO, the computer will wait for you to enter the notes. Type in as many notes as you like from the keyboard. After you have entered about 20 notes, type PRINT. The computer will print and play the complete composition.

Notice that the length of the verse stays the same while the notes move to the left.

```
  1 .COMPOSITION IN L
 10 A=1; INPUT L
 20 NT=5
 30 K=KP
 40 IF K=116GOTO 90
 50 TV=K
 60 @(A)=K
 70 A=A+1
 80 GOTO 30
 90 CLEAR
100 FOR N=1TO A-1-L
110 FOR P=NTO N+L
120 TV=@(P)
130 NEXT P
140 CLEAR
150 NEXT N
```

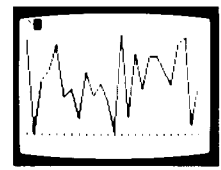
GRAPHS AND CHARTS

These programs draw line and bar graphs. Enter the number of items you wish to graph, then enter the value of each item.



```
  1 .BAR GRAPH
  5 CLEAR
 10 INPUT "◄?►"A
 20 B=150+(A+1)
 30 FOR N=1TO A
 40 PRINT N,
 50 INPUT "?■"@(N)
 60 NEXT N
 70 X=-80+B+2
 80 CLEAR
100 FOR N=1TO A
105 Y=@(N)+2-42
110 BOX X,Y,B+2,@(N),1
120 X=X+B
130 NEXT N
```

LINE GRAPH



```
  1 .LINE GRAPH
  5 CLEAR
 10 INPUT "◄?►"A
 20 B=150+(A-1)
 30 FOR N=1TO A
 40 PRINT N,
 50 INPUT "?"@(N)
 55 IF @(N)►87GOTO 40
 60 NEXT N
 70 X=-80
 80 CLEAR
 90 LINE X,@(1)-44,0
100 FOR N=1TO A
110 LINE X,@(N)-44,1
120 BOX X,-42,1,2,3
130 X=X+B
140 NEXT N
```

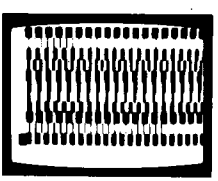
LASER DUEL

Two players cooperate or compete in forming designs as they each move one end of the reverse line.

```
 1  .LASER DUEL
 2.  .BOB OGDON
10  CLEAR
20  X=0; Y=0
30  A=0; B=0
40  X=X+JX(1)×3
50  Y=Y+JY(1)×3
60  LINE X,Y,TR(1)+2
70  A=A+JX(2)×3
80  B=B+JY(2)×3
90  LINE A,B, TR(2)+2
100 BC=KN(1)+5×5
110 FC=KN(2)+5×5
120 GOTO 40
```

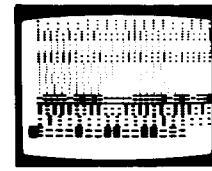
SCROLL ONE

These three programs create electronic paintings that move. The images evolve slowly and the visual experience changes over time.



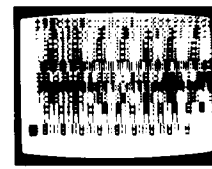
```
 1  .SCROLL ONE
 2  .LARRY CUBA
10  BC=3
20  S=4+RND (4)
30  C=RND (S-3)
40  FOR A=-72TO 77STEP S
50  BOX A,-39,C,8,1
60  NEXT A
70  PRINT
80  FC=7+8×RND (32)
90  GOTO 10
```

SCROLL TWO



```
  1 .SCROLL TWO
  2 .LARRY CUBA
 10 BC=0
 20 S=4+RND (4)
 30 C=RND (S-1)
 40 FOR A=-72TO 77STEP S
 50 T=RND (3)+1
 60 FOR B=-43TO -36STEP T
 70 BOX A,B,C,1,1
 80 NEXT B
 90 NEXT A
100 PRINT
110 FC=7+8*RND (32)
120 GOTO 10
```

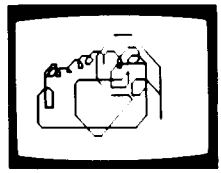
SCROLL THREE



```
  1 .SCROLL THREE
  2 .LARRY CUBA
 10 BC=3
 20 S=4+RND (4)
 30 U=1+RND (3)
 40 C=RND (S-1)
 50 FOR X=-72TO 77STEP U
 60 BOX X,-39,1,8,1
 70 NEXT X
 80 FOR A=-72TO 77STEP S
 90 T=RND (3)+1
100 FOR B=-43TO -36STEP T
110 BOX A,B,C,1,3
120 NEXT B
130 NEXT A
140 PRINT
150 FC=7+8*RND (32)
160 GOTO 10
```

SCRIBBLER

Player one controls the line on the screen. Direction and color of the line are changed by moving the joystick and rotating the knob. The trigger prints the line.



```
 1 .SCRIBBLER
 2 .BOB OGDON
10 CLEAR
20 BC=0
30 FC=KN(1)+5*5
40 X=X+JX(1)*3
50 Y=Y+JY(1)*3
60 LINE X,Y,TR(1)
70 GOTO 30
```

RUBBER BAND

Draw a connect-the-dots pattern on the screen. Moving the joystick controls the direction of the line. Rotating the knob the the right draws a line and rotating the knob to the left leaves a space. The trigger prints each section of the line.

```
 1 .RUBBER BAND
 2 .JAY FENTON
10 CLEAR
20 A=0
30 CLEAR
40 B=0
50 C=0
60 D=0
70 A=A+JX(1)
80 B=B+JY(1)
90 LINE C,D,0
100 LINE A,B,3
110 IF TR(1)GOTO 160
120 LINE C,D,0
130 LINE A,B,3
140 IF KN(1)<0GOTO 170
150 GOTO 70
160 LINE C,D,0;LINE A,B,1
170 C=A
180 D=B
190 IF TR(1)GOTO 190
200 GOTO 70
```

LEARNING SKILLS

LETTER MATCH

When you run the program, four random letters will flash on the screen. Try to repeat the letters exactly. This program automatically adjusts to the player, becoming easier or more difficult to match the player's skill. Get the answers correct and the computer gives you a more challenging game. If you miss, the computer makes the next match easier for you to guess.

```
  1 .LETTER MATCH
 10 S=4
 15 CLEAR
 20 PRINT "CAN YOU REMEMBER"
 30 PRINT S," LETTERS?"
 40 FOR N=1TO S
 50 @(N)=RND (26)+64
 60 TV=@(N)
 70 NEXT N
 80 GOSUB 1000
 90 CLEAR
100 PRINT "GO!"
110 FOR N=1TO S
120 G=KP
130 TV=G
140 IF G#@(N)GOTO 200
150 NEXT N;S=S+1
160 GOSUB 1000
170 GOTO 15
200 PRINT "SORRY"
210 FOR N=1TO S
220 TV=@(N)
230 NEXT N
240 GOSUB 1000
250 S=S-1
260 GOTO 15
1000 FOR T=1TO 500
1010 NEXT T
1020 RETURN
```

ANALOG (NON-DIGITAL) CLOCK

After the clock face appears on the screen the computer will take a few seconds to figure out the coordinates for the minute dots and store them in array locations 0 thru 119. Then, in the upper left corner of the screen you will be asked to INPUT "H", hours, "M", minutes and "S", seconds. When you press GO you'll see the three clock hands, including a moving sweep second hand keeping accurate time. If clock speed needs adjusting change the value of R in line 230. A smaller number will speed up the clock, and a larger number will slow it down.

Notice the missing end-quotes at the ends of lines 90 through 150. They are not required if nothing follows them in a statement. Each character you can eliminate from a program saves one byte that can be used elsewhere in the program for greater enhancement or special effects.

```
  1 .ANALOG (NON-DIGITAL) CLOCK
  2 .BY GEORGE MOSES
 10 NT=0;GOTO 80
 20 D=(C*10)+(B+12*2);XY=0;LINE @(D)+2,@(D+1)+2,3
 30 E=B*2;XY=0;LINE @(E),@(E+1),3
 40 FOR F=A*2TO 119STEP 2;XY=0;LINE @(F),@(F+1),3;FOR T=1TO R;NEXT
    T;XY=0;LINE @(F),@(F+1),3;NEXT F;A=0;XY=0;LINE @(E),@(E+1),3
 50 B=B+1;IF B=60B=0;C=C+1
 60 IF C=12C=0
 70 XY=0;LINE @(D)+2,@(D+1)+2,3;GOTO 20
 80 CLEAR ;FC=140;BC=0;BOX 0,0,160,88,1;BOX 0,0,100,84,3
 90 CX=-2;CY=36;PRINT "12
100 CY=32;CX=-27;PRINT "11",;CX=25;PRINT "1
110 CY=18;CX=-41;PRINT "10",;CX=38;PRINT "2
120 CX=-40;CY=0;PRINT "9",;CX=41;PRINT "3
130 CY=-16;CX=-37;PRINT "8",;CX=38;PRINT "4
140 CY=-32;CX=-24;PRINT "7",;CX=25;PRINT "5
150 CY=-36;CX=1;PRINT "6
160 @(0)=0;@(1)=30;@(2)=4;@(3)=30;@(4)=9;@(5)=30;@
    (6)=13;@(7)=29;@(8)=17;@(9)=28;@(10)=21;@(11)=27
170 @(12)=24;@(13)=25;@(14)=27;@(15)=23;@(16)=29;@
    (17)=21;@(18)=31;@(19)=18;@(20)=32;@(21)=15
180 @(22)=33;@(23)=12;@(24)=34;@(25)=9;@(26)=35;@
    (27)=6;@(28)=35;@(29)=3;@(30)=35;@(31)=0
190 B=28;FOR A=32TO 60STEP
    2;@(A)=@(B);@(A+1)=-(@(B+1));B=B-2;NEXT A
200 B=2;FOR A=62TO 90STEP
    2;@(A)=-(@(B));@(A+1)=-(@(B+1));B=B+2;NEXT A
210 B=28;FOR A=92TO 118STEP
    2;@(A)=-(@(B));@(A+1)=@(B+1);B=B-2;NEXT A
220 FOR A=0TO 118STEP 2;BOX @(A),@(A+1),1,1,1;NEXT A
230 R=466
240 CY=40;INPUT "H"C;CY=40;INPUT "M"B;CY=40;INPUT "S"A;BOX
    -65,40,30,8,1;GOTO 20
```

CHICAGO LOOP

This program incorporates the use of three loops to provide a unique display of graphics looking very much like a city on a lake, complete with reflections, traffic and sound effects.

```
  1 .CHICAGO LOOP
  2 .BY MIKE PEACE
  5 &(16)=70
 10 CLEAR ;FOR A=-80TO 80;BOX A,0,5,RND (95),1;NEXT A
 20 FOR A=1TO 120;B=RND (160)-80;C=RND (40)
 30 BOX B,C,1,1,2;BOX B,C-50,5,1,3;NEXT A
 40 BC=14;FOR A=-80TO 80;BOX A,0,2,1,RND (3);BOX -A,-3,2,1,RND (3)
 50 B=RND (20);C=RND (20);NT=-1
 60 IF B◄14&(21)=0
 70 IF C◄14&(22)=0
 80 IF B►15&(21)=15
 90 IF C►15&(22)=15
100 &(17)=42;&(19)=36;NEXT A;GOTO 40
```

SOUND PORT STUDY

Learn to use the sound ports with this program. JY(1) moves the pointer up and down to select the sound port you wish to play with. JX(1) will fine tune the values you set the port to. Fast, coarse tuning is accomplished by pulling TR(1) while rotating KN(1). Volume 1 controls voices A and B. Volume 2 controls voice C and noise. Note: The symbol, ■, indicates a SPACE.

```
  1 .SOUND PORT STUDY
  2 .BY MIKE PEACE
  6 NT=-1
 10 GOTO 20
 11 PRINT "■■■■M.O.";:RETURN
 12 PRINT "■VOICE A";:RETURN
 13 PRINT "■VOICE B";:RETURN
 14 PRINT "■VOICE C";:RETURN
 15 PRINT "■VIBRATO";:RETURN
 16 PRINT "VOLUME 2";:RETURN
 17 PRINT "VOLUME 1";:RETURN
 18 PRINT "■■■NOISE";:RETURN
 20 CLEAR ;B=15;FOR A=36TO -36STEP -10;B=B+1
 30 CY=A;CX=-65
 35 GOSUB B-5
 40 PRINT #0,"■&(",B,")=
 50 NEXT A;A=5
 60 CX=32;A=A+JY(1)x10;IF A◄-30A=-35
 70 IF A►30A=35
 80 CY=A;B=(A+35)+10
 90 TV=95;TV=31;TV=31
 95 IF JX(1)@(B)=@(B)+JX(1);C=@(B);PRINT #7,C,;&(23-B)=C;GOTO 60
100 IF TR(1)@(B)=&(28);PRINT #7,@(B);&(23-B)=@(B)
110 GOTO 60
```

SIDESWIPE

The car appears on the top of the screen moving toward the bottom. Steer your car using knob (1) to avoid obstacles as they approach. Top score is 100 points. You lose 3 points for each sideswipe and 10 points for each collision.

```
1  .SIDESWIPE
2  .BY MIKE PEACE
3  &(22)=0;&(16)=72;SM=0
4  BC=10;FC=191;&(20)=9;GOTO 7
5  &(18)=9;FOR N=200TO 0STEP -10;&(21)=N;&(22)=N;&(23)=N;NEXT
   N;S=S+10;PRINT "COLLISION";RETURN
6  FOR N=10TO 25;&(22)=255;&(18)=N;NEXT N;&(22)=0;PRINT "SIDE
   SWIPED";S=S+3;RETURN
7  CLEAR ;NT=-1;FOR A=-9TO 0;CX=0;PRINT ".";NEXT A;C=0;S=0
8  BOX -15,0,4,88,1;BOX 15,0,4,88,1
9  PRINT "■■■■■START■■■■■■COURSE";FOR T=1TO 90;E=E+1;IF
   TR(1)RUN
10 R=RND (15);IF R►13Q=-RND (2)
11 IF E►5BOX A+RND (26)-13,-32,6,9,1;E=RND (3)
12 IF A◄-36Q=2
13 IF A►36Q=-2;CX=-72
14 IF A◄-2CX=A+17
15 CX=A;PRINT ".";A=A+Q
16 BOX A-15,-32,4,6,1;BOX A+15,-32,4,6,1;IF A►36Q=-32;CX=-72
17 IF A◄-2CX=A+17
18 IF A◄-36Q=2
19 C=C+KN(1)÷25
20 &(18)=130+ABS(C);&(22)=150
21 BOX C,32,5,7,1;BOX C,31,9,2,1;BOX C,34,9,2,1
22 IF PX(C,27)GOSUB 5;C=A;GOTO 24
23 IF PX(C+4,32)+PX(C-4,32)GOSUB 6;C=A
24 NEXT T;CY=0;CX=-40;PRINT "FINAL SCORE";PRINT #12,100-S
25 &(22)=0;IF H◄100-(S)H=100-S
26 CX=-62;PRINT "TODAY'S HIGH SCORE■",#0,H;PRINT "■■■PULL TRIG-
   GER TO RUN
27 IF TR(1)RUN
28 GOTO 27
```

PERSPECTIVES

This program graphically displays a road going into a city, with telephone poles lining the road. Excellent perspective study!

```
  1 .PERSPECTIVES
  2 .BY MIKE PEACE
 10 BC=111;&(9)=255
 20 CLEAR ;LINE -80,10,0
 30 FOR A=-80TO 80
 40 LINE A,RND (ABS(A)+1)-10,1
 50 NEXT A;LINE 0,-10,0
 60 A=10;FOR C=1TO 20STEP 2;A=A+C;B=A+6
 70 BOX A-3,-B,1+B+2,B x 10,1
 80 BOX A-3,B x 3,B x 5,1+B+2,1
 90 NEXT C;FOR A=-30TO 30
100 LINE A,-44,1;LINE 0,-10,0
110 NEXT A;FOR A=-44TO -11STEP 8
120 BOX 0,A,ABS(A)+10,ABS(A)+6,2
130 NEXT A;A=KP;RUN
```


PERSPECTIVES

This program graphically displays a road going into a city, with telephone poles lining the road. Excellent perspective study!

```
  1 .PERSPECTIVES
  2 .BY MIKE PEACE
 10 BC=111;&(9)=255
 20 CLEAR ;LINE -80,10,0
 30 FOR A=-80TO 80
 40 LINE A,RND (ABS(A)+1)-10,1
 50 NEXT A;LINE 0,-10,0
 60 A=10;FOR C=1TO 20STEP 2;A=A+C;B=A+6
 70 BOX A-3,-B,1+B+2,B x 10,1
 80 BOX A-3,B x 3,B x 5,1+B+2,1
 90 NEXT C;FOR A=-30TO 30
100 LINE A,-44,1;LINE 0,-10,0
110 NEXT A;FOR A=-44TO -11STEP 8
120 BOX 0,A,ABS(A)+10,ABS(A)+6,2
130 NEXT A;A=KP;RUN
```

MAZEMAKER II

Move the + in the desired direction of travel using joystick (1). Time is limited. If you don't make it you may go back and start over. This uses the new two letter Bally BASIC sound-port device variables.

```
  1 .MAZEMAKER II
  2 .BY MIKE PEACE
 10 NT=1;CLEAR;↓;S=0
 20 FOR A=1TO 14;✱(A)=RND (90)+5;NEXT A
 30 BC=RND (32)×8-1;FC=RND (32)×8
100 FOR A=-55TO 70STEP 15;BOX A,0,4,88,1;B=RND (78)-39;BOX
    A,B,4,8,2;NEXT A
105 BOX -55,0,4,10,2
110 FOR A=-40TO 40STEP 20;FOR B=-55TO 70STEP 15
120 IF PX(B,A)BOX B+4,A,5,4,1;MU=B
130 IF PX(B,A+10)BOX B-4,A+10,5,4,1;MU=A
140 NEXT B;NEXT A;&(9)=255;PRINT "■S";PRINT "■T";PRINT "■A";PRINT
    "■R";PRINT "■T";PRINT"►
141 BOX-60,0,6,88,2;BOX 70,0,6,88,2
145 A=-70;B=0;C=0;D=0;S=S+1;FOR T=1TO 300STEP 2
150 IF &(16)C=JX(1)×3;D=JY(1)×3;IF TR(1)RUN
160 A=A+C;B=B+D;BOX A,B,1,3,3;BOX A,B,3,1,3;IF PX(A,B)GOTO 200
170 ↓;IF A►70GOTO 300
175 TB=250-T;VB=15
180 FOR Z=1TO 160-T;NEXT Z;TA=255-T;@(T)=A;@(T+1)=B;IF
    T=252TA=2
190 VA=15;NEXT T
200 FOR Z=20TO 120;TA=Z;TB=Z-6;NEXT Z
210 FOR Z=T-2TO 1STEP -2;A=@(Z);B=@(Z+1);BOX A,B,1,3,3;BOX
    A,B,3,1,3
220 R=Z+14;TA=✱(RM)
230 TB=TA+1;NEXT Z;GOTO 145
300 NT=10;FOR A=48TO 55;MU=A;NEXT
    A;NT=15;MU=49;MU=51;MU=53;MU=62
305 BOX 0,-32,160,12,2;CY=-32
310 NT=0;PRINT "■YOU MADE IT IN■",#0,S,"■TURN",;IF S►1PRINT "S
320 NT=-1;&(22)=0;FOR A=255TO 0STEP -4;&(19)=190;IF TR(1)RUN
330 &(23)=A;&(21)=A;IF A◄4A=256
340 NEXT A
```

WAVEMAKERS FORTUNE TELLER

This program puts a touch of whimsy into computerized fortune telling. There are about 10,000 possibilities. Press any key for another fortune.

The symbol: ■, means SPACE

```
 1   .WAVEMAKERS FORTUNE TELLER
 2   .BY MIKE PEACE
 5   CLEAR;NT=0;CY=20
 6   FOR A=10TO 40STEP 10;GOSUB A+RND (10);NEXT A
 7   FC=RND (32)x8;BC=FC-81
10   A=KP;RUN
11   PRINT "■YOU WILL WITHOUT A DOUBT";PRINT "■MEET",;RETURN
12   PRINT "■YOU ARE GOING TO CRASH";PRINT "■INTO",;RETURN
13   PRINT "■YOU WILL START WORKING";PRINT "■FOR",;RETURN
14   PRINT "■YOU CAN EXPECT A SUPRISE";PRINT "■FROM",;RETURN
15   PRINT "■YOU WILL LEARN A SECRET";PRINT "■FROM",;RETURN
16   PRINT "■YOU'RE THE KIND OF";PRINT "■PERSON THAT NEEDS";RETURN
17   PRINT "■YOU WILL BECOME";RETURN
18   PRINT "■YOU WILL APPEAR IN A";PRINT "■BROADWAY PLAY
     AS";RETURN
19   PRINT "■YOU'LL RECEIVE ONE";PRINT "■MILLION DOLLARS
     FROM";RETURN
20   PRINT "■YOU WILL BECOME INVOLVED";PRINT "■WITH";RETURN
21   PRINT "■A HANDSOME STRANGER";RETURN
22   PRINT "■A LARGE ELEPHANT";RETURN
23   PRINT "■A RICH EXECUTIVE";RETURN
24   PRINT "■A T.V. REPAIRMAN";RETURN
25   PRINT "■A HAIR STYLIST";RETURN
26   PRINT "■A SOPHISTICATED TYPE";PRINT "■PERSON";RETURN
27   PRINT "■A TRUCK DRIVER";RETURN
28   PRINT "■A BELLY DANCER";RETURN
29   PRINT "■AN OLD LOVER";RETURN
30   PRINT "■A STRANGE MIDGET";RETURN
31   PRINT "■WITH A WART ON THE NOSE";RETURN
32   PRINT "■IN A PHONE BOOTH";RETURN
33   PRINT "■WITH LOTS OF MONEY";RETURN
34   PRINT "■ON DRUGS";RETURN
35   PRINT "■ON A TRAIN";RETURN
36   PRINT "■WITH A LIMP";RETURN
37   PRINT "■THAT RAN OUT OF GAS";RETURN
38   PRINT "■300 POUNDS OVERWEIGHT";RETURN
39   PRINT "■WHO IS 89 YEARS OLD";RETURN
40   PRINT "■ON STILTS";RETURN
41   PRINT "■TONIGHT";RETURN
42   PRINT "■TRY TO BE COOL";RETURN
43   PRINT "■NEXT MONTH";RETURN
44   PRINT "■SO LIVE IT UP!";RETURN
45   PRINT "■ISN'T IT WONDERFUL?";RETURN
46   PRINT "■WITH A TRAINED CHICKEN";RETURN
47   PRINT "■EXPECT THE UNEXPECTED";RETURN
48   PRINT "■IN LESS THAN 24 HOURS";RETURN
49   PRINT "■YOU COULD DO WORSE!";RETURN
50   PRINT "■BUT DON'T WORRY ABOUT IT";RETURN
```

CHARACTER SET SIZE MULTIPLIER

This program uses POKE & CALL to generate character sets using multiplication factors of 2x, 4x or 8x. This program POKES a small machine language program (converted to decimal) to call up On-Board Subroutine #52 (String Display Routine). This program, along with many others in these pages was submitted to us by Fred Cornett, publisher of The Basic Express, a monthly newsletter serving Bally BASIC hobbyists.

After inputting the program and pressing RUN and GO, the computer will print "LETTER SIZE?" and then "INPUT 2, 4, OR 8". The computer is asking you to input what size letter you wish to display; 2, 4 or 8 times normal size. After you input your selection press "GO"; the screen will clear and wait for you to input characters. Try the following. Input "HI". After you have input the last character, press GO. The screen will clear and display letters in the new size. When ready to start over, press any key.

NOTE: Be careful when using the 8x size. If you put in more characters than will fit on the screen, the program will bomb!

```
10  CLEAR
20  PRINT "LETTER SIZE?
30  INPUT "INPUT 2, 4 OR 8"L
40  IF L=2L=18456;GOTO 80
50  IF L=4L=-26600;GOTO 80
60  IF L=8L=-10216;GOTO 80
70  GOTO 20
80  CLEAR ;M=19975;N=M;G=210
90  P=-39;GOSUB G
100 P=53;GOSUB G
110 P=L;GOSUB G
120 P=19985;GOSUB G
130 P=-13863;GOSUB G
140 M=19985
150 C=KP;TV=C
160 IF C=13GOTO 180
170 %(M)=C;M=M+1;GOTO 150
180 %(M)=0
190 CLEAR
200 CALL (N);C=KP;GOTO 10
210 %(M)=P;M=M+2;RETURN
```

DISPLAY 256 COLORS AT ONCE!

This program has especially delighted the average user as it appears to be impossible to accomplish such a thing with Bally BASIC... You will find out how good your television is as this program will push its color displaying capabilities to the limit! Thanks to The Basic Express newsletter for sharing this program with us.

```
  1  .256 COLOR DISPLAY
  2  .BY JERRY BURIANYK
 10  A=20200;B=A;C=400;CLEAR
 20  X=-9741;GOSUB C
 30  X=20030;GOSUB C
 40  X=18413;GOSUB C
 50  X=-2754;GOSUB C
 60  X=3539;GOSUB C
 70  X=-1063;GOSUB C
 80  X=-2102;GOSUB C
 90  X=-12978;GOSUB C
100  X=20117;GOSUB C
110  X=-4621;GOSUB C
120  X=8819;GOSUB C
130  X=12622;GOSUB C
140  X=20002;GOSUB C
150  X=-14859;GOSUB C
160  X=-6699;GOSUB C
170  X=-6691;GOSUB C
180  X=-6659;GOSUB C
190  X=7387;GOSUB C
200  X=211;GOSUB C
210  X=467;GOSUB C
220  X=723;GOSUB C
230  X=979;GOSUB C
240  X=15637;GOSUB C
250  X=-3040;GOSUB C
260  X=-7683;GOSUB C
270  X=-7715;GOSUB C
280  X=-11807;GOSUB C
290  X=-3647;GOSUB C
300  X=31725;GOSUB C
310  X=20002;GOSUB C
320  X=-13829;GOSUB C
330  &(15)=255;&(9)=18
340  CALL (B);GOTO 500
400  %(A)=X;A=A+2;RETURN
500  FOR E=0TO 255STEP 8
510  BC=E;FC=E-2;CY=40;CX=-77
520  PRINT E;NEXT E
```

ARTILLERY DUEL

Artillery Duel is an intriguing game submitted courtesy of The Arcadian, a monthly newsletter serving the Bally BASIC programming hobbyist and published by Bob Fabris. This program sets up a random mountain scene and adds two gun emplacements. As each player's turn is taken, he adjusts the knob for barrel elevation, moves the joystick to add or reduce the number of gunpowder bags (by whole bags sideways; by tenths back and forth). Then when ready, pull the trigger. There is gravity and a random wind. The gun recoils and fires the shell. There is an explosion when it lands. A gun is destroyed when less than half a gun remains (the repair crew can replace a gun barrel). The program uses all available space , so don't enter lines 3 and 4. Be sure to exercise the joystick to see how the variables work.

```
  3 .ARTILLERY DUEL
  4 .BY JOHN PERKINS
 10 FOR A=0TO 9;@(A)=0;NEXT A
 20 CLEAR ;H=RND (50)-44;A=-80;T=RND (9);LINE A,H,4;I=H
 30 A=A+RND (10);H=H+RND (9)-T;IF H◄-44H=-44;T=4
 40 IF H►-10T=6
 50 T=T+(T►5)×2-RND (3)+1;IF A►78A=79
 60 LINE A,H,1;IF A#79GOTO 30
 70 H=I;W=RND (51)-26
 80 FOR A=-80TO 79;LINE A,-44,4;IF PX(A,H)GOTO 120
 90 FOR Y=1TO 60;IF PX(A,H+Y)H=H+Y;GOTO 120
100 IF (PX(A,H-Y)=0)+(H-Y◄-44)NEXT Y
110 H=H-Y
120 LINE A,H,1;NEXT A;S=RND (2)×2-3;G=RND (3)+2
130 FOR P=0TO 5STEP 5;X=RND (31)-75+(P=5)×120;FOR H=-44TO 20;IF
    PX(X,H)NEXT H
140 Y=H+RND (5)-3;IF Y◄-40Y=-40
150 BOX X+(P=0)×4-2,Y+4,11,15,2;BOX X,Y,5,5,1;BOX
    X+(P=0)×2-1,Y-3,3,1,1
160 LINE X,Y,4;LINE X+(P=0)×12-6,Y,1;@(P+1)=X;@(P+2)=Y
    ;NEXT P
170 P=(S=1)×5;C=P+5+1
180 FOR N=1TO 11;BOX @(P+1),@(P+2)+1,3,1,3;NEXT N;NT=1
190 W=W+RND (9)-5;CX=-9;CY=40;PRINT "WIND";CX=-9;IF W
    TV=95+(W►0)×2
200 PRINT #3,ABS(W)
210 CX=S×51-22;CY=40;A=@(P+4);PRINT "ANGLE",#3,A
220 CX=S×51-22;B=@(P+3);PRINT "BAGS■",#1,B+10,".",RM
230 IF TR(C)GOTO 350
240 K=(KN(C)+128)×S+(S#1)×255;IF ABS(K-E)◄10GOTO 300
250 E=K;A=K+15×5;@(P+4)=A;CY=40;CX=S×5
    1+8;PRINT #3,A
260 GOSUB 280;X=@(P+1);Y=@(P+2);LINE X-3×S,Y,4;BOX
    X-5×S,Y+4,5,9,2
270 K=K+100;J=RM;LINE X-(3+K+25)×S,Y+J+25,1;GOTO 300
280 GOSUB 500+A×2×(A◄45)+(90-A)×2×(A►40);IF
    A►45K=K+100+RM×100
290 RETURN
300 IF JX(C)=0IF JY(C)=0GOTO 230
```

```
310 B=B+JX(C)*10+JY(C);IF B<0B=0
320 IF B>99B=99
330 @(P+3)=B;CX=S*51+8;CY=32
340 PRINT #1,B+10,".",RM;GOTO 230
350 BOX 0,36,159,16,2;BOX @(P+1),@(P+2)+1,3,1,1
360 GOSUB 280;X=K+100;Y=RM;R=9
370 X=-(X*B+100)*S+W+2;Y=Y*B+100
380 I=(@(P+1)-3*S)*10;J=@(P+2)*10;NT=-1;FOR N=15TO 1STEP
    -1;&(21)=32;&(23)=N*16;NEXT N
390 U=I+10;V=J+10;BOX U,V,1,1,3
400 K=U;L=V;Y=Y-G;I=I+X;J=J+Y;U=I+10;V=J+10;BOX K,L,1,1,3;BOX
    U,V,1,1,3;IF ABS(U)>79BOX U,V,1,1,2;U=99;GOTO 430
410 IF V<-40V=-40;GOTO 430
420 IF (PX(U,V))+(V>20)GOTO 400
430 LINE U,V,4;FOR N=240TO 0STEP -16;&(23)=N;BC=  ND (3)*13+86
440 LINE U+RND (R)-R+2-1,V+RND (R)-R+2,2;NEXT N
450 BC=7;T=0;E=(S#1)*5;U=@(E+1);V=@(E+2);FOR X=U-2TO U+2;FOR
    Y=V-2TO V+2;IF PX(X,Y)T=T+1
460 IF T>13S=-S;GOTO 170
470 NEXT Y;NEXT X;IF R=9R=19;GOTO 430
480 @(P)=@(P)+1;PRINT #5,@(0),"■■■■DESTROYED",@(5)
490 FOR N=0TO 3000;NEXT N;GOTO 20
500 K=9900;RETURN
510 K=9908;RETURN
520 K=9817;RETURN
530 K=9625;RETURN
540 K=9334;RETURN
550 K=9042;RETURN
560 K=8650;RETURN
570 K=8157;RETURN
580 K=7664;RETURN
590 K=7070;RETURN
```

COUNT THE BOXES

Here is a child's game that teaches counting in a fun way. It compliments you when you're correct and invites you to try again when you're wrong. Enter skill level from 10 (easy) to 100 (hard).

The symbol ■ means SPACE

```
 1 .COUNT THE BOXES
 2 .BY BRETT BILBREY
10 CLEAR ;R=0;W=0
20 INPUT "■SKILL LEVEL?"D
30 IF D<10PRINT "THAT'S TOO EASY";FOR L=1TO 1000;NEXT L;GOTO 10
40 IF D>100PRINT "THAT'S TOO HARD";FOR L=1TO 1000;NEXT L;GOTO 10
50 D=110-D
```

```
 60  CLEAR ;C = 0;BC = RND (255);FC = BC + 4;FOR A = - 70TO 70STEP (RND
     (D) + D + 5 x 2)
 70  BOX A,RND (60) - 20,4,4,1
 80  C = C + 1
 90  NEXT A
100  CY = - 30;PRINT "HOW MANY BOXES CAN YOU SEE",
110  INPUT "YOUR GUESS?■■■■"G;IF G#CGOTO 150
120  NT = 2;FOR L = 1TO 7;MU = 55;BOX  - 1, - 32,159,20,3;NEXT L;BOX
      - 1, - 32,159,20,2;NT = 0
130  CY = - 30;PRINT "GOOD";D = D + 1;IF D►100D = 100
140  R = R + 1;PRINT "YOU HAVE■",#0,R,"■RIGHT";FOR L = 1TO 1000;NEXT
     L;GOTO 60
150  NT = 3;FOR L = 1TO 7;MU = 10 - L;BOX  - 1, - 32,159,20,3;NEXT L;BOX
      - 1, - 32,159,20,2;NT = 0
160  CY = - 30;PRINT "TOO BAD";D = D - 1;IF D◄10D = 10
170  W = W + 1;PRINT "YOU HAVE■",#0,W,"■WRONG";FOR L = 1TO 1000;NEXT
     L;GOTO 60
```

DECIMAL TO HEX CONVERTER

This short program will take any decimal number between 0 and 15 and display its hexidecimal counterpart. The program's greatest value is in its illustration of the use of Boolean Logic. That is, the use of mathematical expressions in parentheses which, if true, have a value of 1 and, if false, have a value of 0. These values, in line 30, influence the outcome of the screen display because they are multiplied and added to determine the value of "TV."

This program would be valuable as a subroutine inserted in a larger program working with hexidecimal numbers.

```
 1  .DECIMAL TO HEX CONVERTER
 2  .BY JEFF FREDERIKSEN
 3  CLEAR
10  PRINT "A = ",;INPUT ""X
20  CY = CY + 8;CX = ↑X + 42
30  TV = (X + 48) x (X◄10) + (X + 55) x (X►9) x (X◄16)
40  PRINT ;GOTO 10
```

APPENDIX A

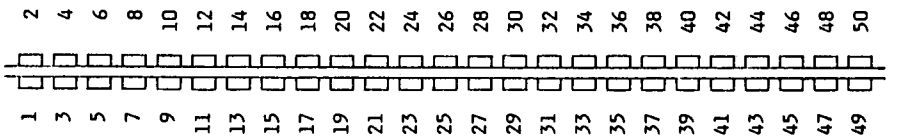
CHARACTER CODE TABLE

Code #	Char.	Code #	Char.	Code #	Char.	Code #	Char.
0	??	32	SPACE	64	@	96	↓
1	??	33	!	65	A	97	→
2	??	34	"	66	B	98	× (multiply)
3	??	35	#	67	C	99	+
4	??	36	$	68	D	100	graphic
5	??	37	%	69	E	101	graphic
6	??	38	&	70	F	102	graphic
7	??	39	' (apostrophe)	71	G	103	graphic
8	??	40	(	72	H	104	LIST
9	??	41	)	73	I	105	CLEAR
10	??	42	*	74	J	106	RUN
11	??	43	+	75	K	107	NEXT
12	??	44	,	76	L	108	LINE
13	GO	45	– (minus)	77	M	109	IF
14	??	46	. (period)	78	N	110	GOTO
15	??	47	/	79	O	111	GOSUB
16	??	48	0	80	P	112	RETURN
17	??	49	1	81	Q	113	BOX
18	??	50	2	82	R	114	FOR
19	??	51	3	83	S	115	INPUT
20	??	52	4	84	T	116	PRINT
21	??	53	5	85	U	117	STEP
22	??	54	6	86	V	118	RND
23	??	55	7	87	W	119	TO
24	??	56	8	88	X	120	??
25	??	57	9	89	Y	121	??
26	??	58	:	90	Z	122	??
27	??	59	;	91	[	123	??
28	??	60	<	92	\	124	??
29	??	61	=	93	]	125	??
30	??	62	>	94	↑	126	??
31	ERASE	63	←	95	←	127	??

APPENDIX B

BUS AND CONNECTOR STRUCTURES

Astro expansion bus connector pin configuration:



Signal Definitions

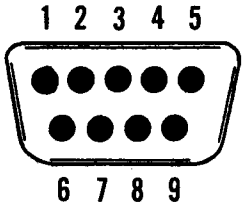
1. GND.	18. A6	35. D6
2. VIDOUT	19. A2	36. A1
3. VIDIN	20. A4	37. D2
4. GND.	21. A12	38. A0
5. 7.16 MHZ	22. A3	39. D0
6. I/O 1.79 MHZ	23. A11	40. D7
7. PX (3.58 MHZ)	24. A15	41. $\overline{\text{NMI}}$
8. AUDIN	25. A10	42. D1
9. *$\overline{\text{MI}}$	26. A13	43. $\overline{\text{BUSRQ}}$
10. VERT. DR.	27. A9	44. **BUSAK
11. $\overline{\text{RESET}}$	28. A14	45 *$\overline{\text{WR}}$
12. HOR. DR.	29. *$\overline{\text{RFSH}}$	46. $\overline{\text{WAIT}}$
13. *$\overline{\text{IORQ}}$	30. A8	47. CASEN
14. *$\overline{\text{RD}}$	31. D4	48. $\overline{\text{HALT}}$
15. A7	32. $\overline{\text{INT}}$	49. $\overline{\text{BUZOFF}}$
16. *$\overline{\text{MREQ}}$	33. D5	50. SYSEN
17. A5	34. D3	

NOTE: * Indicates signals tri-stated by **.

SPECIAL ASTRO SIGNALS

CASEN—Disables cassette ROM port.
SYSEN—Disables system ROM.
BUZOFF—Disables custom chips, leaves ROM and CPU.

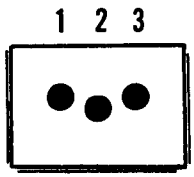
HAND CONTROL PIN CONFIGURATION



SIGNAL DEFINITIONS

1. Trigger
2. Right
3. Left
4. Down
5. N.C.
6. 50K Pot
7. Ground
8. 50K Pot (End)
9. Up

LIGHT PEN CONNECTOR PORT PIN CONFIGURATION



SIGNAL DEFINITIONS

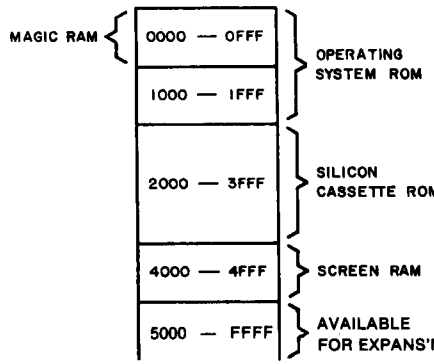
1. Ground
2. $\overline{LP}$
3. +5 Volts

NOTE: Improper use of any connectors may damage internal circuits. Proper grounding is necessary to prevent static damage. If you use these connectors for your own devices, you must take responsibility for any damage done to your computer by those devices.

APPENDIX C

MEMORY MAP

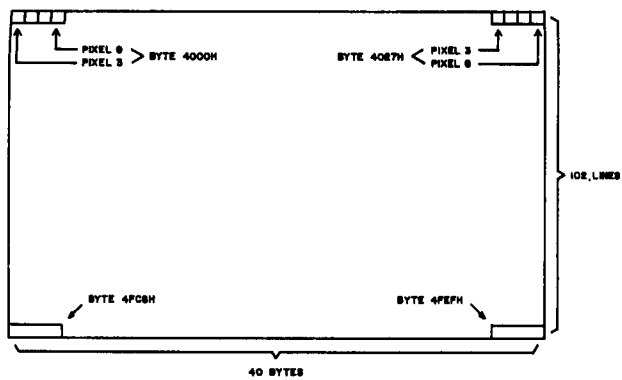
SYSTEM MEMORY MAP



MAGIC RAM

This is a function where an attempt to write to this ROM will result in a modified form of the data being placed into memory at the location attempted plus 4000 Hex, and modified by the contents of the magic register.

SCREEN MEMORY MAP



Two bits of RAM define one screen pixel. One byte then contains 4 screen pixels.

APPENDIX D

INPUT AND OUTPUT PORTS

INPUT PORTS

HEX	FUNCTION
08	Intercept Feedback Register: Used with screen writes to determine if data has been written on top of non-zero pixel values.
0E	Vertical Addr: For use with light pen.
0F	Horizontal Addr: For use with light pen
10	Switch Bank 0: Joystick switch 1.
11	Switch Bank 1: Joystick switch 2.
12	Switch Bank 2: Joystick switch 3.
13	Switch Bank 3: Joystick switch 4.
14	Switch Bank 4: Keypad column #4 (rightmost).
15	Switch Bank 5: Keypad column #3.
16	Switch Bank 6: Keypad column #2.
17	Switch Bank 7: Keypad column #1 (leftmost).
1C	Potentiometer 0: Value of Knob 1
1D	Potentiometer 1: Value of Knob 2
1E	Potentiometer 2: Value of Knob 3
1F	Potentiometer 3; Value of Knob 4

OUTPUT PORTS

HEX	FUNCTION
00	Color Register 0
01	Color Register 1
02	Color Register 2
03	Color Register 3
04	Color Register 4
05	Color Register 5
06	Color Register 6
07	Color Register 7
	Color Registers 0—3 are used for pixels mapped to the right of the port 9 boundary. Registers 4—7 are used for pixels mapped to the left of the port 9 boundary. In BASIC ports 4 & 5 are set to BC and ports 6 & 7 are set to FC.
08	Sets consumer/commercial mode for custom chips.
09	Sets left/right color mapping boundary. Bits 6 & 7 map the border color.
0A	The # of lines $\times 2$ to display from top of screen. (204 maximum).
0B	Color Block Transfer Register
0C	Magic Register. Determines how to modify screen data.
0D	Stores vector for use with IM2 interrupts.
0E	Sets type of interrupt mode in custom chips
0F	Sets number of lines to scan before generating an interrupt.
10	Tone Master Oscillator
11	Tone A
12	Tone B
13	Tone C
14	Vibrato
15	Tone C volume
16	Tone A & B volume
17	Noise volume
18	Sound Block Transfer Register

All ports are accessed using the &( ) construct. You can only write to an output port. You can only read from an input port.

A = &(X) means set the variable A equal to the most recently sampled value of port X.

&(X) = A means set port X equal to the value stored in variable A.

APPENDIX E

BALLY BASIC DATA BASE LOCATIONS

	HEX	DECIMAL
Bally BASIC ROM area	2000—2FFF	8192—12287
Graphics/program area	4000—4E10	16384—19983
Scratchpad area	4E20—4FEA	20000—20458
TXTUNF (end of BASIC program address)	4E20	20000
Variables start at:	4E22	20002
Stack area	4F22—4FEA	20258—20458
Text/Array area	A000—A70C	−24576 to −22777
Line input buffer	4EBA—4F21	20154—20257
HOOK VECTORS		
Light pen interrupt	4E92	20114
Screen interrupt	4E95	20117
Input character	4E98	20120
Output character	4E9B	20123
Stack vector (2 bytes)	4E9E	20126

Many of the new Bally BASIC's data base locations differ from the original version of Bally BASIC introduced in 1978. This means that some of the machine language programs written in the old Bally BASIC will not run in the new Bally BASIC unless the locations for data storage and CALL routines are changed to correspond to the new BASIC's memory layout.

APPENDIX F

300 BAUD TO 2000 BAUD TAPE CONVERSION PROGRAM

When the original version of Bally BASIC was released in 1978 a separate interface had to be plugged into the number three hand controller port to output the program to cassette tape. It was much larger and slower than your new built-in interface, and it took as long as four minutes to load a long program from tape to memory. The speed was only 300 baud (bits per second). Compare that to your new BASIC with its 2000 baud interface which can load the longest program in under 20 seconds!

While the new BASIC has many added features, the language has remained almost exactly the same. This means that hundreds of programs written for the old Bally BASIC will run on the new BASIC. The only problem is, the old 300 baud tapes won't load directly into the new interface. This leaves you with the choice of manually typing them into your Arcade with the new BASIC in place, or using this program that Jay Fenton provided to put the new interface into a mode that will accept the old 300 baud tapes. This is a machine language program that will take some time to build. It involves using this intermediate "Array Builder" program to input 231 different values into consecutive array addresses and using our directions to produce your own tape of this special utility program.

The first step is to type in this program:

```
  1  .ARRAY BUILDER
  2  .BY JAY FENTON
 10  CLEAR ;INPUT "START"A
 20  PRINT "@(",#0,A,
 30  INPUT ")="@(A)
 40  A=A+1
 50  GOTO 20
100  INPUT "START"A
110  PRINT "@(",#0,A,")=",@(A)
120  A=A+1
130  GOTO 110
200  :PRINT @(0),231
```

RUN this program. The screen will clear and will ask you to input the starting address of your array. We want to start at @(0) and build an array to @(230). So type in a zero and press GO. The computer will now prompt you for each consecutive value by printing the array address followed by an equals sign and waiting for your input. When the array address is printed on the screen, type in the number appearing to the right of that address on the following list. For example, if the computer prints:

@(0)=

Consult the list, then type:

-20493

Then press GO and look for the next number. Be careful to input a minus sign when you come to a negative number on the list. Every number has to be correct for this program to work, so, proofread everything.

@(0) = −20493
@(1) = 2515
@(2) = 5843
@(3) = 574
@(4) = 467
@(5) = −11513
@(6) = 15874
@(7) = −11513
@(8) = 15875
@(9) = −11316
@(10) = 8458
@(11) = 17024
@(12) = 10274
@(13) = −9393
@(14) = −6633
@(15) = 8255
@(16) = −9455
@(17) = −22763
@(18) = 6338
@(19) = 833
@(20) = 3960
@(21) = 3855
@(22) = 2022
@(23) = 211
@(24) = −5864
@(25) = −11345
@(26) = 11776
@(27) = −13046
@(28) = 16593
@(29) = −1752
@(30) = 8237
@(31) = −12808
@(32) = 16529
@(33) = 15102
@(34) = 6952
@(35) = 27390
@(36) = 8138
@(37) = −12991
@(38) = 16511
@(39) = −4304
@(40) = −30515
@(41) = −12992
@(42) = 16529
@(43) = 32717
@(44) = 14400
@(45) = −267
@(46) = 10272
@(47) = 6388
@(48) = −13039
@(49) = 16529
@(50) = 8446
@(51) = −1752
@(52) = 28926
@(53) = −10720
@(54) = 7107
@(55) = −12992
@(56) = 16529
@(57) = 8190
@(58) = −1752
@(59) = −30515
@(60) = −448
@(61) = 10253
@(62) = 6338
@(63) = −272
@(64) = 14384
@(65) = −509
@(66) = −10182
@(67) = −13913
@(68) = 10282
@(69) = 30543
@(70) = 8739
@(71) = 20264
@(72) = −20535
@(73) = 5843
@(74) = −21555
@(75) = 8256
@(76) = 8696
@(77) = 2048
@(78) = −11396
@(79) = −13034
@(80) = 16593
@(81) = 4021
@(82) = 9583
@(83) = −2272
@(84) = 4563
@(85) = 3785
@(86) = 7680
@(87) = −9472
@(88) = −22763
@(89) = 6338
@(90) = 14913
@(91) = 15360
@(92) = 14935
@(93) = 15360
@(94) = 4010
@(95) = 1336
@(96) = 8220
@(97) = 6390
@(98) = 31462
@(99) = −31917
@(100) = 4094
@(101) = −7880
@(102) = 257
@(103) = 6144
@(104) = 265
@(105) = 0
@(106) = 5595
@(107) = −15705
@(108) = 16664
@(109) = 30
@(110) = 58
@(111) = 22332
@(112) = 58
@(113) = −21956
@(114) = 14351
@(115) = 7173
@(116) = −2528
@(117) = −6888
@(118) = 58
@(119) = −21956
@(120) = 12303
@(121) = 7173
@(122) = −2528
@(123) = −9960
@(124) = −389
@(125) = 14348
@(126) = 3085
@(127) = −392
@(128) = 12294
@(129) = 30994
@(130) = 1278
@(131) = −11720
@(132) = −13905
@(133) = 30980
@(134) = 1022
@(135) = −2000
@(136) = −392
@(137) = 14344
@(138) = −20539
@(139) = −14020
@(140) = −5583
@(141) = 3919
@(142) = 7122
@(143) = −20672
@(144) = 5843
@(145) = 211
@(146) = 29473
@(147) = −4799
@(148) = 10331
@(149) = 335
@(150) = 5
@(151) = −20243
@(152) = 4587
@(153) = 20054
@(154) = 6955
@(155) = 13950
@(156) = 4608
@(157) = −388
@(158) = 8258
@(159) = 32245
@(160) = −32514
@(161) = −4064
@(162) = 21485
@(163) = 20264
@(164) = 8234
@(165) = 8782
@(166) = 20262
@(167) = 21793
@(168) = 8789
@(169) = 16386
@(170) = 1057
@(171) = 8864
@(172) = 20000
@(173) = 10785
@(174) = 8783
@(175) = 20121
@(176) = 8995
@(177) = −25566
@(178) = 8526
@(179) = 16759
@(180) = 10769
@(181) = 335
@(182) = 86
@(183) = −20243
@(184) = 12739
@(185) = 3365
@(186) = 3433
@(187) = 6399
@(188) = −7401
@(189) = 31989
@(190) = 11518
@(191) = 3360
@(192) = −387
@(193) = 8296
@(194) = −3832
@(195) = −14879
@(196) = −10779
@(197) = 28867
@(198) = −3796
@(199) = −13853
@(200) = 8234
@(201) = −6834
@(202) = 9770
@(203) = 8783
@(204) = 20000
@(205) = 10282
@(206) = 32335
@(207) = 54

@(208) = 8739	@(216) = 15618	@(224) = 8587
@(209) = 20264	@(217) = −55	@(225) = −20418
@(210) = 8234	@(218) = 8731	@(226) = 2771
@(211) = 8782	@(219) = 13390	@(227) = 11326
@(212) = 20262	@(220) = 0	@(228) = 2515
@(213) = 8929	@(221) = 24575	@(229) = 12739
@(214) = 20000	@(222) = 20114	@(230) = 37
@(215) = 10300	@(223) = 14	

When you have entered the last number on the list, press HALT. Next, enter the proofreading mode of the program by typing GOTO 100. The computer will again ask you for the starting address. Type a zero and press GO. The computer will proceed to list the array contents from @(0) thru @(230) in order. Carefully check them against the list printed here, inspecting them also for their positive or negative status.

When you are certain the entire array is in order you are ready to instruct the computer to generate the machine language program to tape. Place a blank tape in your cassette recorder and cue it beyond the five second leader if it has one. With your recorder running on RECORD, type GOTO 200. The computer will then write the array to tape. When the cursor reappears, repeat the procedure, making two or three copies of the program for safekeeping.

After you have made your tape, a method of checking to see if the data is accurate is to RESET the computer and type:

:INPUT @(0)

Load the program from the tape into the computer. When the cursor reappears type:

PRINT @(231)

The computer should print 58 on the screen. If it is something else, you have made a mistake.

OPERATING INSTRUCTIONS:

Bally BASIC has a special command for loading machine language programs directly into screen memory, bypassing the BASIC language format entirely. It is called "COLON-RUN," and is typed in similar to the familiar :INPUT command. RESET your computer and type:

:RUN

Load the TAPE CONVERTER PROGRAM you just built, using the :RUN command. When loaded, the screen background brightness will change continuously. Cue up the 300 baud tape, press GO, and load the 300 baud program into the computer. Observe the program statements accumulating in the buffer just below the program zone. The screen will reveal the scratchpad during this phase.

Program loading will stop on any of the following conditions:

(a) :RETURN is found in an unnumbered line.
(b) RUN is found in an unnumbered line.
(c) Any other key in the column beneath HALT is pressed.

Cases (a) and (c) will cause the program to go back to the flashing grey scale mode. Case (b) will cause the program to proceed to final loading. If you wind up in the grey scale mode and have no more segments to load, press HALT to get to the final loading. After HALT is pressed, or RUN is found, the program will convert the statements in the buffer into the representation used within your Bally BASIC. This is done by moving the buffer to the highest possible address and using special I/O routines which feed characters from this buffer to the Bally BASIC input routines.

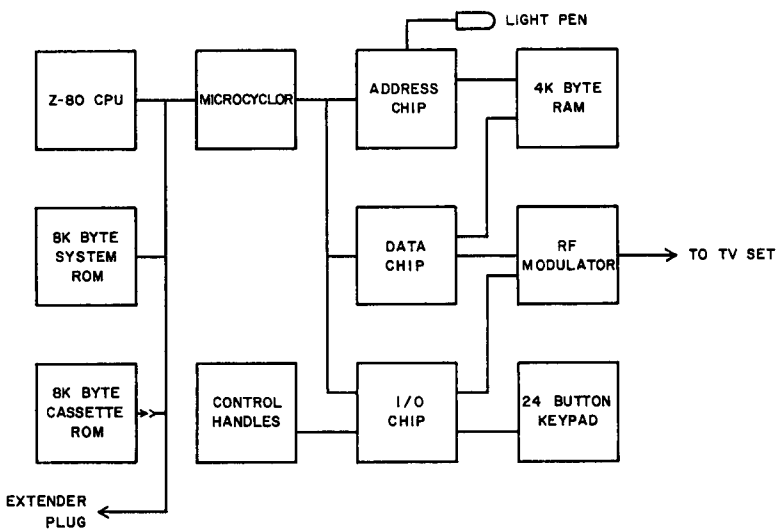
This will be seen on the screen as a race between the input buffer and the program storage area. The CONVERTER PROGRAM will then exit to Bally BASIC. The variables A to Z will be zeroed out, and the screen will be cleared. The program may now be executed or written out at 2000 baud.

NOTE: During the read-in phase, lines without line numbers, rubout characters, and spaces appearing before and immediately after line numbers are filtered out. It is permissible to load tapes which contain line numbers out of sequence or which delete previous lines.

JAY FENTON

APPENDIX G

SYSTEM BLOCK DIAGRAM



SYSTEM BLOCK DIAGRAM

APPENDIX H

BALLY BASIC COMMANDS AND FUNCTIONS SUMMARY

BASIC STATEMENTS AND COMMANDS

BOX X,Y,A,B,1

means draw a box in foreground color that is centered at the point X,Y. The box is A pixels wide and B pixels high. You can draw boxes in four modes:

BOX X,Y,A,B,1 foreground color box
BOX X,Y,A,B,2 background color box
BOX X,Y,A,B,3 reverse box
BOX X,Y,A,B,4 invisible box

CLEAR

means clear the screen. Any program stored in memory remains unchanged.

FOR/TO/STEP/NEXT

These commands form a loop.

```
10FOR A = 1TO 16STEP 3
20PRINT A,
30  NEXT A
```

This loop prints 1, 4, 7, 10, 13, 16. The variable A is set to 1. The values 16 and 3 are stored, as well as the name of the variable, A. Execution continues until a NEXT command is encountered. The STEP can be positive, negative, even zero. If STEP is omitted it defaults to a value of +1. The value of A is checked with that of the most recent FOR command. Then A is set to its current value plus the value of the STEP expression. The new value of A is checked to see if it exceeds the TO expression. If not, execution will return to the command following the FOR command. If the value of A exceeds the limit, the loop is finished.

IF

means test the value of an expression. If it is not zero (if it is true) the rest of the statement will be executed. IF A = 5GOTO 20 means if the number stored in A is 5 the expression is true and has a value of 1. If it is not 5, the expression equals zero, is false and the rest of this statement will be skipped. Execution will continue at the next numbered statement.

INPUT A

tells the computer to stop and wait for you to enter a number. The variable named on the screen (A) is set equal to your keypad entry when you press GO.

INPUT "HOW MANY?"A

means print "HOW MANY?" on the screen and then set A equal to the number you input.

LINE X,Y,1

means draw a line to the point X,Y in foreground color.
You can draw lines in four modes:
LINE X,Y,1 line in foreground color
LINE X,Y,2 line in background color
LINE X,Y,3 reverse line
LINE X,Y,4 invisible line.
Each line is drawn beginning at the end point of the most recent line drawn. The location of this end point is stored in the two letter variable, XY. You can begin a line elsewhere on the screen in two ways: Draw an invisible line (mode 4) to the desired new beginning point and proceed from there, or type in XY = A, where the value of A is the desired relocation point of the XY coordinate.

LIST

will print on the TV all instructions in numerical order beginning with the lowest line number.

LIST ,5

means start at the beginning and print out only the first five numbered statements.

LIST 100

means start with line number 100 and list to the end.

LIST 100,5

means start with line 100 and list only five numbered statements (inclusive).

PRINT "A"

means print the character A on the screen.

PRINT A

means print the value of A on the screen.

PRINT #A,B

means leave A spaces and print the value of B. Omitting the #A, Bally BASIC will normally print numbers right-justified in an 8 space field.

RETURN

means return to the instruction following the most recent GOSUB. The computer remembers which GOSUB to return to. A RETURN must be the last command in a line.

RND (A)

generates a random number between one and the value of A.
If A = 0 the range is from −32768 to 32767.
If A is between 1 and 32767, RND (A) will generate a number between 1 and A, inclusive.

RUN

means run the program after you press GO, beginning with the lowest statement number. To begin elsewhere in the program use a GOTO followed by the line number you wish to start at.

SPACE

means leave a space on the screen. Spaces don't matter to the computer, but they can make your instructions easier for you to read. Numbers and command words cannot have imbedded spaces. A space is required to separate two similar symbols in an IF statement:

```
WRONG   10IF A = BC = 100
RIGHT   10IF A = B  C = 100
```

IMMEDIATE COMMANDS

RESET

completely clears memory, erases any program, sets all variables and arrays to zero.

ERASE

means remove the last command or character you typed on the screen. This doesn't work after you have pressed the GO or HALT key.

GO

means enter the instructions typed on the screen into memory. If GO follows a command word it means execute the command. Press GO after each instruction.

GO + 10

is used by pressing the WORDS shift key before pressing GO. It tells the computer to enter the last line, add 10 to the line number and print the sum as a new line number to begin the next statement.

GENERAL FUNCTIONS

The following functions are not included in the words on your keypad overlay. To use these functions each letter must be typed separately.

ABS (A)

is the absolute value of the number stored in the variable A.
PRINT ABS(−44)
prints 44, the absolute value of −44 on the screen. ABS can be used to get a positive value from an algebraic expression:
PRINT ABS(X × Y + 60 ÷ N)

CALL (A)

transfers control to an assembly language subroutine at address A. This routine must be Z-80 machine code and terminated by a RET instruction. Also, registers must be exchanged when entering and exiting the routine if you plan to return to BASIC.

RM

contains the remainder of the most recently executed integer division.

SM

sets scroll mode by:

SM = 0 Normal scrolling
SM = 1 No scrolling. Cursor stays at bottom of screen
SM = 2 Holds cursor at screen bottom. Clears after each carriage return
SM = 3 Clears the screen and resets cursor to screen top
SM = 4 AUTO-PAUSE, release by pressing any key. After release, the screen will clear and printing will resume from screen top.

STOP

causes your program to stop. 100STOP will stop your program at line 100.

SZ

is the number of unused memory locations. PRINT SZ tells the computer the print this number on the screen.

XY

stores the XY position specified in the latest LINE command. The Y value occupies the high order byte of this word, X the lower byte.

H

means halt the program and return control to you. Halt is an immediate command and cannot be typed into a program. (See STOP.)

PAUSE

means stop the computer and wait. You can pause while running or listing a program. Press any key to continue.

TRACE

holding down the blue and gold shift keys while the program is running prints statements about to be executed, then executes them.

INPUT-OUTPUT FUNCTIONS

JX(1)

is the function that gives the position of the number one joystick horizontally.

Left	JX(1) = −1
Center	JX(1) = 0
Right	JX(1) = 1

JY(1)

is a function that gives the position of the number one joystick vertically.

Forward	JY(1) = 1
Center	JY(1) = 0
Back	JY(1) = −1

TR(1)

is a function that tells the status of the number one trigger.

Pulled	TR(1) = 1
Not Pulled	TR(1) = 0

KN(1)

is a function that gives the position of the number one knob. The range is from −128 counterclockwise to 127 clockwise.

A = KP

means wait until you press a key on the keypad. Each key has a number and the number of the key you press is stored in the A counter. You can see the ASCII code of the key you pressed with the instruction TV = A.

TV = A

means put a letter or other character on the TV. The character is the one that matches the number stored in A. See KP.

MU = A

means play a note in the TV speaker that matches the number in the A counter.

MU = "A"

means play a note in the TV speaker that's the same as the note you hear when you press the letter A.

FC

is the number of the foreground color.

BC

is the number of the background color.

NT

is the note time. After RESET the note time is set to two, or two-sixtieths of a second. The larger the note time the longer the tones will last when each key is pressed. NT = 0 will turn off the note time.

CX

is the number that places the cursor left or right on the screen. The range is from −80 to 79.

CY

Is the number that places the cursor up or down on the screen. The range is from 43 to −44.

TAPE COMMANDS

:PRINT

Save RAM (program/variables/screen) to tape.

:INPUT

Load program from tape to memory.

:LIST

Provides checksum of program stored on tape without affecting program in memory. Prints a question mark to left of cursor if an error is found.

:RUN

The :RUN command is provided for loading machine language programs. The load will begin at the top of the screen (4000H or 16384D). When the load is completed control will be transferred to this first address. The block loaded is limited in size only by the need to avoid interference with the stack area.

:PRINT @(0),100

To write out blocks of data, a second form of :PRINT is provided. This will write a data block of indicated length beginning at the specified address. For example, to write out the first 100 words of the @( ) array, :PRINT @(0),100 will save array addresses @(0) thru @(99) on tape. Arrays are discussed in detail in Lesson 4.

:INPUT @(0),100

As with :PRINT there is a form of :INPUT for loading data blocks. :INPUT @(0),100 will specify that the load is to begin at array address zero and continue loading a total of 100 addresses.

ERROR MESSAGES

WHAT?

The computer says WHAT? when it doesn't understand you.

SORRY

The computer says SORRY! when there isn't enough room in its memory to do what you want.

HOW?

The computer asks HOW? when it understands what you want but cannot figure out how to do it.

PUNCTUATION AND OPERATORS

; The semi-colon means the same thing as a line number. It separates multiple statements on the same line.

, The comma means continue. In PRINT A, the comma after A means continue printing on the same line.

. The period after a line number creates a REM statement. That is a statement that you can use to make a remark to the user of your program. The computer will ignore anything in a statement after the period, unless the period is in a print statement between quotes.

► means "is greater than," as 5►3.

◄ means "is less than," as in 8◄12.

= means "is equal to."

means "not equal to."

B = %(A)
means PEEK into memory location A and store the value found there in the variable B.

%(A) = B
means take the value stored in variable B and POKE it into memory location A. (For a detailed Bally BASIC memory address map see Appendix E.)

B = @(A)
means store the value found in array address A in the variable B.

@(A) = B
means take the value found in the variable B and store it in array address A.

***( )**
is the second array symbol and can be used instead of @( ). It allows you to store data at the end of the free memory space instead of at the beginning, as with the @( ) array, avoiding the possibility of data being lost if you expand the size of your program into that memory space.

B = &(A)
means read the value in port A and store it in the variable B.

&(A) = B
means take the value found in variable B and write it to port A.

↓
is the command to turn off all of the sound ports when they are being addressed by the two-letter music processor variables (discussed later in this appendix and also in Lesson 9.)

ARITHMETIC

Your computer is designed to work the multiplication and division portions of a problem first, and the addition and subtraction portions last.

$3 \times 5 - 2 = 13$ (not 9)
Parenthesis will change this order.
$3 \times (5 - 2) = 9$ (not 13)
Whole numbers only are used.
$15 \div 2 = 7$ (not 7½)

MUSIC PROCESSOR COMMANDS

A set of new device variables have been provided to facilitate access to the Bally Arcade music synthesizer. The synthesizer can be divided into two sections. The first section is concerned with controlling the master oscillator. The master oscillator output is inputted to the other section which contains the 3 voice oscillators. Thus changes made to the master oscillator affect all 3 voices.

The master oscillator is a programmable frequency divider. It is a counter which is clocked at 1.789 Mhz. Each time it counts down to zero, the state of its output is toggled and the counter is reinitialized to the value set in MO. The master oscillator frequency output is a square wave of frequency FM = 1789 Khz ÷ (MO + 1).

By changing the value of NM (NOISE MODE) the behavior of the master oscillator may be modified. Setting NM = 1 causes noise to be added to MO. This sum is used to reset the counter in the master oscillator. The effect is that the frequency is varied by a random amount. The amount of variation is controlled by the variable NV (NOISE VOLUME).

When NM = 0, vibrato is enabled. Vibrato works like noise modulation, except that the value added to MO varies over a programmable range 0—63. This range is controlled by VR. The rate at which this added value varies is determined by the vibrato frequency register VF. VF can have four different values: 0 is fastest, 3 is the slowest.

The right side of the synthesizer consists of three frequency dividers (voices) with associated volume controls. Each divider is clocked by the output of the master oscillator. The output frequency is given by the formula: FV = FM ÷ (2(Tn) + 1).

Where Tn is TA, TB, or TC, for voices A, B, and C respectively. Substituting in the formula for FM we have: FV = 894 Khz ÷ (MO + 1)(Tn + 1).

Each voice has a 4 bit volume control variable. Zero is quiet, 15 is full volume. The volume is linearly proportional to the value of the variable. VA corresponds to TA, VB to TB, and VC to TC.

White noise can be mixed in with the output of voices A, B, and C by setting NM = 2. The volume of this noise is determined by NV. Setting NM = 3 sends noise both the MO and to final output.

A few notes about using these variables:

1. When NT►0 printing characters on the screen will change the values of MO, TA, and VA. This effect can be disabled by setting NT = 0.
2. Changes to these variables will be propagated 60 times a second, like FC and BC.
3. The direct use of the sound ports &(16) thru &(23) will conflict with using these variables. To turn off these variables set NT = −1. This will be necessary when running programs developed in the old Bally BASIC which program the sound ports. For your information the sound ports are arranged as follows:

&(16) master oscillator	&(20) vibrato
&(17) oscillator A	&(21) noise control × 6 + vol C
&(18) oscillator B	&(22) volume B × 16 + volume A
&(19) oscillator C	&(23) noise volume

To silence the music processor when using the ports it is necessary to set each port to zero. This can be done fastest by using a for/next loop from 16 to 23.

Silencing the music processor when using the new variables is delightfully easy. Just use the down arrow symbol. This can be used in a program as a statement, or from the keypad as a direct command. NT must equal zero or the down arrow will have no effect.

APPENDIX I

GENERAL SPECIFICATIONS BALLY ARCADE

MICROPROCESSOR	Z-80
MEMORY	
RAM	4K Bytes
ROM	8K Bytes
ROM	8K Bytes (max. from cartridge)
INPUTS	
Calculator Keypad	24 Keys
Knobs	4 (50 K pots)
Joysticks	4
Triggers	4
Light pen	Provision
OUTPUT GRAPHICS	
Resolution	16,320 pixels
Configuration	160 pixels wide, 102 pixels high
Display	Color or black and white
Number of colors	256

OUTPUT AUDIO
1 Channel triple tone with tremelo and vibrato

OUTPUT SIGNAL
NTSC standard color

OUTPUT RF CHANNELS
3 or 4

POWER CONSUMPTION
12 WATTS (avg.)

POWER REQUIREMENTS
120 VAC standard

ADDITIONAL SPECIFICATIONS WITH BALLY BASIC

LANGUAGE
Palo Alto Tiny BASIC

CASSETTE TRANSFER RATE
2000 BPS

OUTPUT TEXT	
Display	286 Characters
Capacity	26 char./line by 11 lines
OUTPUT GRAPHICS	
Resolution	14,080 pixels
Configuration	160 pixels wide, 88 pixels high

Bally® BASIC

BALLY BASIC

LIMITED WARRANTY

Astrovision, Inc., 6460 Busch Blvd., Suite 215, Columbus, OH, 43229 (the "Warrantor") hereby warrants, to the original purchaser only, that this product will be free from defects in materials and workmanship, under normal use, for a period of 90 days from the date of purchase.

The Warrantor shall have no liability or responsibility to purchaser or any other person or entity with respect to any liability, loss or damage caused or alleged to be caused directly or indirectly by this product, including but not limited to any interruption of service, loss of business and anticipatory profits or consequential damages resulting from the use or operation of this product.

If during this 90-day period a defect in this product should occur, the product may be returned to: Astrovision, Inc., or to an authorized Astrovision, Inc. dealer and Astrovision, Inc. will replace this product without charge.

When requesting performance under the terms of this warranty, the original purchase date must be established by the customer by means of a bill of sale, invoice, or other acceptable documentation.

This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Some states do not allow the exclusion or limitation of incidental or consequential damages, so the above limitations or exclusions may not apply to you.

ASTROVISION INC.

6460 BUSCH BLVD., SUITE 215
COLUMBUS, OHIO 43229

MADE IN U.S.A. THE IMAGE PRODUCERS, INC. PRINTED IN U.S.A.